

# Chapter 1 – Introduction

As we make use of technologies to enhance our lives and perform feats that were not possible on our own, these systems being used are continuously improved upon to have more intelligence, be efficient in performing the given tasks, and also reducing the resource costs in the development and maintenance processes.

One particular area that is currently growing in interest is the field of robotics. The scope of this field encompasses a large range of disciplines, as the systems and techniques that are developed can be used to assist in comprehending and automating the activities that have previously required human interventions. Although traditional applications of robots have been focused on non-reactive and repetitive tasks, there has been a push towards developing interactive systems that can adapt their behaviour depending on immediate and historical interactions with the users and the environment. This process is carried out through on-board sensors or external sensor systems that can relay a specific set of states of the environment depending on the type of sensors used and how they are used (Dudek & Jenkin, 2000).

The interactive systems provide a platform for the implementation of a wide variety of algorithms and techniques, as the sensors are typically constructed to be generic devices for capturing the data while the processing algorithms differ between applications. The price of these sensors and the ease in integration has also become an important issue to allow not just the research institutes, but hobbyists and scientists of different disciplines to expand into and take from the field of mobile robotics.

## 1.1 Mobile robotics

The majority of robotic systems that are currently used in today's world have predefined tasks they carry out at specific locations, thus the interactions require physically moving the object of interest within the range of the robotic systems. This can restrict the type of tasks it can perform, especially if mobility is restricted. Although restricted mobility can provide benefits such as predictability, controlled scenarios, and safety, the ability to manoeuvre allows the interactive tasks to be carried out in a much wider variety of locations, which greatly increases the type of tasks the robots can be assigned. The primary task for mobile robots typically involves the replacement of biological systems to carry out repetitive, strenuous, or hazardous tasks either fully autonomously or with partial manual intervention from a remote location (Buhmann et al., 1995; Burgard et al., 1998; Horchler et al., 2003; Mayer, 2001; Sibley et al., 2002; Tucakov et al., 1997; Yamauchi et al., 1998; Zlatev & Balkenius, 2001).

The techniques and algorithms that are used depend on the tasks and the availabilities of the sensors included with the mobile robot systems, which may be specifically designed for the task, or generic sensors that are used in a certain way to enhance the capabilities of the robots. The development of the robot itself, the sensor

## 1.1 Mobile robotics

usage, and the algorithms for processing the sensory data are all integral part of the system which can be developed in parallel or separately, before being combined together. Due to the cross-over between different applications where the sensors and algorithms can be re-used, many algorithms are developed independently for different purposes and are later integrated with the mobile robot system (Hu & Gan, 2005; Rajendran & Huber, 2004; Shen et al., 1998).

The flexibility in being able to attach multiple components means the physical configuration of the mobile robot can differ significantly between systems. This also includes consideration to the placement of the non-sensory components such as the locomotive components, housing of the processing unit, providing power to run the mechanical components, as well as the overall size of the robot, which all depend on the tasks set for the robot and the resources that are available to construct the robot.

An alternative approach is to define the tasks by observing the configuration of an existing mobile robot platform (Eklundh et al., 1996). This approach is more common, as the cost of developing a dedicated system is often not plausible until the capabilities of the robot has been fully defined. An interchangeable design allows portions of the system to be modified to suit particular tasks without the need to replace the common components, thus allowing experimental development to be carried out with ease.

The specific tasks carried out by the mobile robot can be split into four major phases. These are the observation of the environment by the sensors, the processing of the sensor data to interpret the current states of the robot and the surroundings, the adaptation of the internal states with regard to the goal of the system, and finally the response by the mobile robot to interact with the environment. Each of these phases can be further broken down into more specific processes that are the focus for researchers within the field of robotics. Figure 1.1 illustrates the phase cycle, where the decisions made by the robot are dependant on the current and historical information, as well as the overarching purpose of the system. This generic cycle represents the high level flow of process for interactive or closed-loop systems.

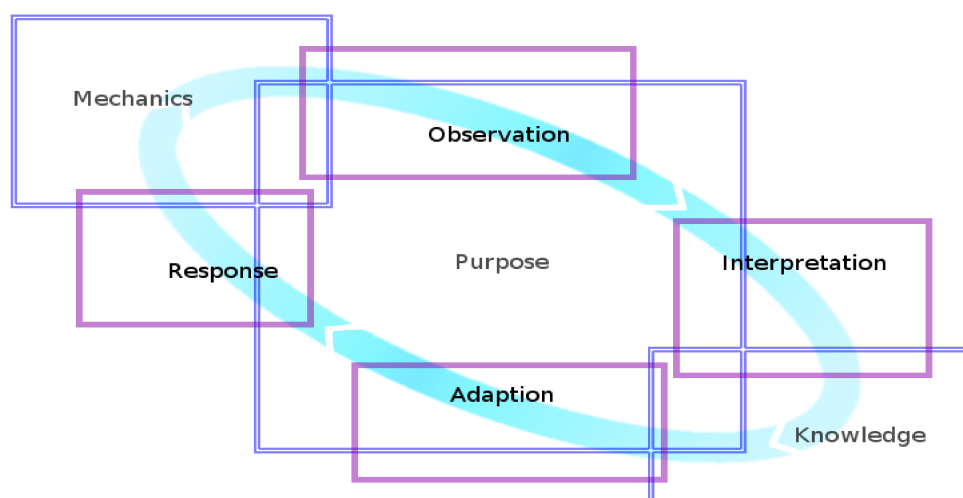


Figure 1.1: Process cycle for a mobile robot.

The double lined categories represent the components or constraints of the system, while the solid lined categories represent the processes and communication between them.

## 1.2 Simultaneous localisation and mapping

### 1.2 Simultaneous localisation and mapping

A commonly seen theme throughout mobile robotics is the notion of recognising the pose changes of the robot (Borenstein et al., 1997; Kleinberg, 1994; Zimmer, 1995), as well as merging and comprehension of the historical sensory information to form a model of the environment (Debevec, 1996; Thau, 1997). The two tasks are strongly coupled since the pose changes are typically determined by the changes in the perspective of the scene structure while the map construction requires the precise pose changes to be combined with the sensor readings about the surroundings (Betge-Brezetz et al., 1995; Burgard et al., 1996 (a); Csorba, 1997).

The localisation aspect can range significantly; The such as by simply maintaining mechanical estimates of the position changes to a more relativistic model that encode the poses with respect to the surroundings. The ability to relate its pose to the actions being performed allows the task to be carried out at multiple locations and be differentiated. Although it is possible to operate without knowing the current location of the robot, assuming or fixing the location can restrict its capabilities, especially if the robot is to interact with a dynamic environment.

The techniques that are used for localisation fall within two basic categories, where one makes use of the domain knowledge about the locomotive behaviour and configuration (Kelly & Murray, 1994; Ostrowski, 1999), called open loop, while the other, called closed-loop, makes use of the sensory feedback about the current state of the environment and the robot. The first approach is commonly seen in situations where the environment and the robot system is well modelled or as an assistance measure to monitor the difference between anticipated action and the actual actions.

The second approach is one that has attracted much research interest, as there is an enormous number of ways to combine various sensors and to interpret the state of the surroundings to disambiguate the current pose of the robot (Huang et al., 2005; Ishiguro & Tsuji, 1996; Jensfelt, 2001). These techniques typically involve triangulation processes using distinctive observations, probability based approaches to indicate the confidence in the various states (Bouguet & Perona, 1995; Bulata & Devy, 1996; Chin & Dyer, 1986; Davison et al., 2007; Dellaert et al, 1999; Fox et al., 2001; Thrun, 2000; Thrun et al., 2001), or correlation process between multiple expected and measured models of the environment (Eklund et al., 1994; Eliazar & Parr, 2003; Mandelbaum, 1995; Wijk et al., 1997).

To be able to perform the correlation, as well as being able to inform the other systems of the state of the environment, a virtual model of the environment can be constructed to allow the maintenance of historical sensory information at various poses. The construction of these models or maps often involves a number of sensors with varying modality to observe the surroundings through multiple view points. The process consists of interpreting the sensor measurements to reconstruct the environment by extracting the distinguishable components and overlaying the measurements from multiple view points to disambiguate and accurately locate the objects into the model. (Roy & Dudek, 2001; Thrun, 1998).

The interpretation of the sensor signals and the integration of multiple measurements are often synonymous to many non-robot based research, as the technology behind the algorithm can easily be interchanged with slightly different

## 1.2 Simultaneous localisation and mapping

objectives. These include fundamental algorithms such as search and clustering techniques to more specific algorithms like object tracking and energy minimisation algorithms to find optimal solutions.

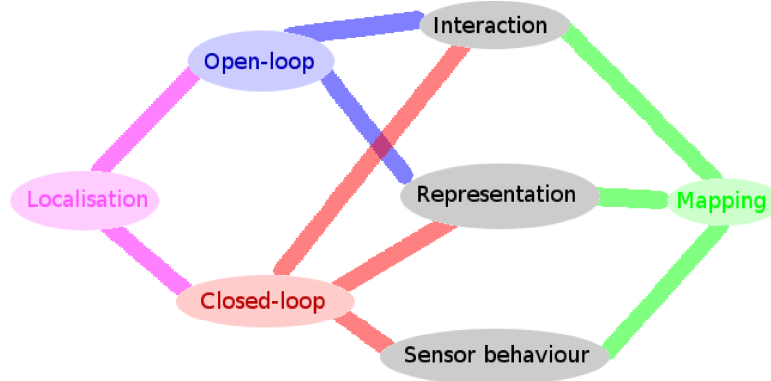


Figure 1.2: Interactions between localisation and mapping. The difference between the open and closed-loop approaches, as well as the interactions with the mapping can be seen.

As figure 1.2 shows, the two components of Simultaneous Localisation and Mapping (SLAM) are dependant on each other to provide disambiguation by the sharing of information and the use of the sensors. The integration between the two areas typically involve iteratively processing the sensor readings from the two perspectives and making use of the continual stream of sensor scans of the same object to gradually decrease the errors in the representation. This design allows the individual components to be developed more independently and merged later on to improve each other. Since the mobile robot often has an exploration component to its behaviour, the reliability and accuracy of the two components is crucial as any errors that are introduced can quickly propagate and have a snowball effect on the accuracy of the internal states.

## 1.3 Overview

The overarching theme behind this work is on the development of techniques and algorithms for an indoor modelling mobile robot using affordable sensors. The focus has been placed on the software side for adaptability of the proposed approaches on other platforms and disciplines using off-the-shelf hardware that can be easily integrated. The use of multiple sensors provides disambiguation and alternate perspective for reliability and accuracy in the models that are constructed. However, the integration and simultaneous processing of multiple modules means that one of the primary limitations is resource consumption, which is constantly dealt with in each of the algorithms.

Each chapter contains introductory and background information on the area, while more specific details of existing work are included throughout the body of the thesis when they are directly referred.

This thesis is organised into three sections illustrating the sequence of development in the various components of the mobile robot platform. Section 1 contains three chapters; locomotion, sensors, and processing, which cover the three



### 1.3 Overview

key subcomponents of the platform.

The overview of the system defines the capabilities and limitation of the robot. This guides the tasks that can be carried out, as well as configuration knowledge that can be exploited when designing the algorithms that interpret the data.

The base localisation technique and the configuration of the sensors involved are discussed in section 2. Discussion on the configuration process is described in chapter 5, while chapters 6 and 7 focus on the localisation technique and the integration of multiple sensors to achieve high precision pose maintenance.

The focus of this section is in the development of an accurate and fast local localisation technique. The closed-loop approach deals with problems encountered with traditional dead reckoning approaches, such as slippage and inappropriate motion models being used. The improvement in the accuracy and reliability in the local localisation implementation means it can be used to enhance global localisation techniques. The proposed approach allows for reduced burden on meeting the correct criteria of finding multiple distinguishable features to correct the pose, such as when exploring new areas, as well as reducing the frequency of pose corrections to reset drifting errors.

While implementing the above techniques, several related problems were encountered and resolved, such as the noise reductions and synchronisation between multiple features. The noise reduction filters that have been developed use the camera characteristics to distinguish between the intended sensor reading and noise. The majority of these filters are thus applicable in other fields which make use of image streams.

The feature synchronisation plays an important role in the accuracy of many models that are used to translate the discrete sensor readings. Rather than reducing the level of precision that is derived for the feature pose, the sub-unit characteristics are determined, which allows simpler interaction between features, as well as maintaining the continuity in the sensor readings.

Lastly, section 3 covers the mapping aspect of the system and the high level interpretations of the environment through the image sensors. Chapter 8 covers the basics of the mapping process while chapter 9 introduces the mapping algorithm used to construct the model of the environment. Chapters 10 and 11 introduces various strategies to enhance both the localisation and mapping processes through the use of high level constructs to improve the efficiency of the sensor measurement interpretations.

In this section, the focus shifts towards a fast and meaningful interpretation of the sensor readings. The fast carving algorithm, together with the orientation dependant range finders to map interaction, removes the limitation on the scan frequency. This also allows the precision used in the grid map to be increased and the use of other modules simultaneously due to the reduced processing load.

The algorithms dealing with the high level interpretation of the sensor data have mostly been customised for the purpose of enhancing the attributes that are maintained in the map. This includes the criteria and considerations for feature recognition for identifying and tracking surface boundaries, the grouping of such features to improve the pose triangulation, and recognising the presence of dynamic

### 1.3 Overview

objects to notify the map of possible changes.

The image processing algorithms observe the characteristics of objects seen through the cameras to select the appropriate sampling points and the maintenance strategies while being mindful of the processing load.

Several other techniques are proposed, but are currently not fully integrated with the other modules. This includes a landmark detection algorithm to observe the change in floor texture patterns, surface segmentation as an alternative approach to the surface boundary detection, and some optimisation approaches to the mapping process.

Finally, the thesis is concluded with a summary and general discussion on the project in chapter 12. A more detailed list of contributions is also described here.

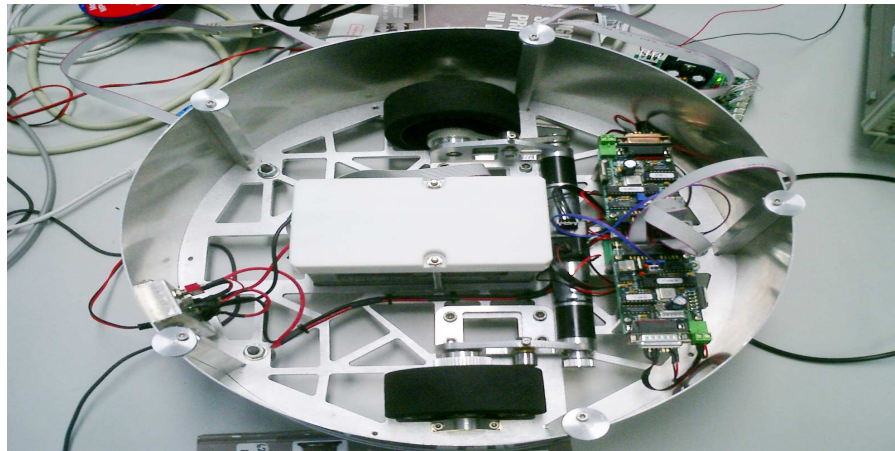


Figure 1.3: Mobile robot platform base.  
The base includes the battery and the locomotive components.

The mobile robot platform used throughout this project started off as a simple base, as shown in figure 1.3, which has now developed into a multi-tiered module and sensor carrier, as shown in figure 1.4. The incremental attachment of the various modules and sensors are discussed throughout the thesis.

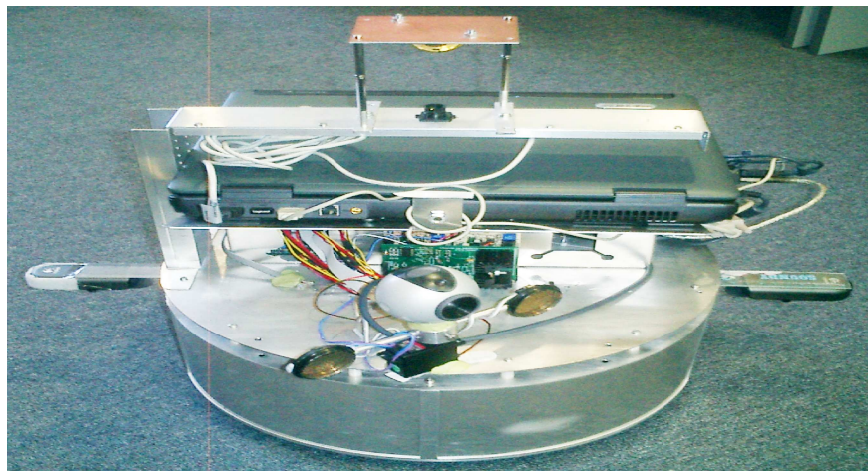


Figure 1.4: Current mobile robot platform.  
The current implementation includes five sensor modules that were incrementally added to enhance the robot's capabilities.

## Section 1 – Platform

*“Like the symbiosis between the mind and the body, the platform provides the constraint and means for the algorithms to operate.”*

A common approach to developing mobile robots is done by clearly distinguishing between specific modules that take care of a specific task. In most cases, one is developed after the other with small amounts of constraints and optimisation being applied during the integration process. This style of approach allows for incremental development with solutions to specific tasks. However, some of the components cause interference or do not allow for simple integration due to the lack of foresight. This characteristic is different for purpose built robot systems, which are very efficient and capable in using the existing devices, but typically come complete and do not allow additional components due to the high level of coupling between the various components. This can limit the adaptability as it becomes very difficult to add or derive new capabilities for the robot system (Brooks, 1991).

The platform which has been used for this project is designed to be an extensible and simple interfaced system to be developed in incremental phases to allow the development and addition of extra modules as undergraduate student projects (Arnold, 2004; Fonseca, 2007; Nagchaudhuri, 2002). This requirement causes the physical size of the robot to increase over time as additional modules are attached. Fortunately, due to the nature of most projects, each module mainly focus on one type of sensor to allow clean distinction of the functionality of modules and a simple interface between the processor and the sensors.

The two basic shapes typically used on mobile robot systems are rectangles and circles. Although the decision in selecting the shape is typically based on the locomotive mechanisms, there are additional considerations such as the compactness, sensor arrangement, and also the nature of the tasks to take into consideration. The rectangular shapes often allow more compactness due to the shapes of typical mechanical systems and also provide balanced ground contact control, thus it is often used for outdoor systems where ground coverage plays a significant role in the robot's functionality. For robots which require higher degrees of control in motion such as an omnidirectionally sensing systems, circular robots are more commonly employed due to the uniform interface to the environment for many of the sensors.

Instead of chaining a series of carriage like modules behind the robot, the extension modules are typically placed on top of each other in a towering fashion (Ostrowski et al., 1997). The major benefit of the chaining structure is that it allows the individual modules to control the elevation of the sensors quite freely, but it can hinder the motion behaviours from the extra points of contact (Borenstein, 1993) and also the operational direction of the mounted sensors caused by obstruction. Integration between the modules becomes a difficult task due to the high level of coupling required to anticipate the locations of the other modules. By stacking each of the modules on top of each other, there is a constant constraint placed on the configuration and introduces a much simpler dependency between the modules at the cost of limited positions of the sensors.

With the above in mind, a circular and layered platform was built. The footprint of the robot was made large enough to house a wide variety of additional sensors and devices which would be included in the future, but small enough to allow operation in a confined indoor environment. The bottom base measures at 400 mm in diameter with a height of 130 mm to house the components that does not need extensive external access, such as some circuit boards for communicating with sensors, the battery, motors for controlling the wheels, as well as the majority of the wheel itself.

One of the other significant physical design constraints was with regards to whether the robot will be tethered or not. It is highly desirable for a mobile robot to operate completely untethered to allow more flexibility in the environment they can be used in (Feng et al., 1996). For this to occur, a 7 A h splash proof / gell-cell lead-acid battery pack was installed in the base of the robot, as well as a laptop computer mount to allow higher level program designs and implementation to be carried out on the robot itself. It can also allowing the off-loading of some processing tasks to external systems in the future due to the simple networking capability between a laptop and another computer. The laptop and battery contributed for the majority of the weight of the robot, which measures at approximately 7 kilograms with no other load.

This section covers other physical issues which govern the mobile robot, as well as the characteristics of various sensors that are present. Chapter 2 covers the issues of locomotion, chapter 3 focuses on the characteristics of the currently available sensors, while chapter 4 will discuss the processing issues as well as covering the communication issues between the various modules.

## Chapter 2 – Locomotion

The definition of the term *robot* can vary between sources and environments, but one of the fundamental qualities of a robot is its ability to interact with the environment. A common approach is by physical interactions with external systems through a medium that is typically beyond the robot's control. The components involved in this type of interaction include how it will change the environment by touching and manoeuvring itself and the object to change its pose. This behaviour has strong links with the study of locomotion, which deals with the notion of self-propelled motion to change its pose.

Depending on the deployed environment of the robot, the type of motion and the mechanical requirements can vary significantly (Halperin et al., 2004). Although a wheeled robot base is commonly seen for terrain robots, there are many other types like pedal and self-rolling robots (Pratt et al., 1997), as well as an equally diverse range of motion inducers that exist for other mediums, such as propeller and wing powered robots for underwater and aerial exploration. The popularity of the wheel based robots on land is mainly due to its simplicity in the implementation and modelling the motion, as well as being an efficient form of motion in terms of power usage.

The modelling process of a single driving wheel vehicle is quite simple, as the motion is dependent on the direction of the wheel and the circumference of the wheel. Although this simplified model does not consider issues such as ground and wheel compaction, traction, lateral slip, and backlash, the approximation can be used to predict and plan the robot's motion with reasonable accuracy. Figure 2.1 illustrates some of these characteristics that are often ignored in simplified motion analyses.

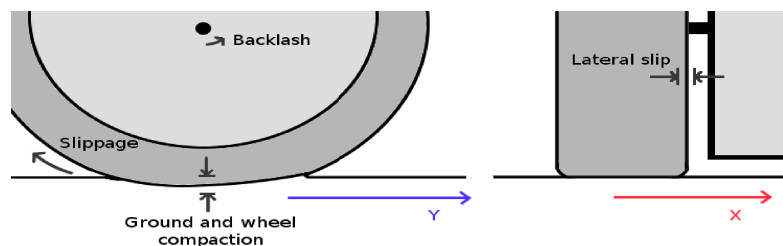


Figure 2.1: Various components of a wheel based locomotive system. This illustrates many of the wheel characteristics that are often ignored in motion models, thus leading to inaccuracies and drifting errors.

To balance the robot, additional wheels are often placed to increase the ground contact points. These wheels, which are called castor wheels, do not provide any driving force or steering functionality and are simply there as support. These are often freely rotating in any axes, thus are left out in most motion modelling calculations. Unlike the training wheels on a bicycle, the castor wheels, which can be seen in figure 2.2, typically do not hinder the motion due to the constant traversal on a single plane.

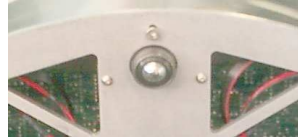


Figure 2.2: Castor wheel.

The robot's balance is maintained by three castor wheels located around the robot.

By increasing the number of driving wheels, the robot is able to traverse to a given point with more control and variety in the path. This is achieved through a combination of forward and backwards traversals, sideways motion, as well as rotations that are centered on artificial pivot points, which is sometimes referred to as the instantaneous center of curvature (ICC). This pivot point is derived from the intersecting point between the rotational axes of the wheels, which can be altered by changing the orientation of one or more steering wheels with respect to each other, or by modifying the relative velocities of the wheels, which will be described below.

## 2.1 Differential drive

Differential drive systems are equipped with two or more driving wheels that are individually controlled by motors, such that they lie along a common axis. This allows the center of curvature to occur anywhere along that axis, which allows rotation to occur by varying the relative motor speeds of the wheels. For example, by setting the velocity of one motor to be the opposite to the other, but at the same magnitude, the robot will rotate around the midpoint of the two wheels. This type of motion allows controlled motion in confined locations without the risk of collision.

This configuration is one of the simplest and commonly implemented approaches, that allows for a large variety of motion to occur and is suited for a smooth terrain environment with many obstacles. However, due to the high sensitivity to the relative velocity between the wheels, the smallest amount of difference can result in a change of trajectory. It also relies on a smooth lateral slippage to occur, thus place an extra emphasis on the synchronisation between the two wheels and knowing the exact motions taken by the wheels.

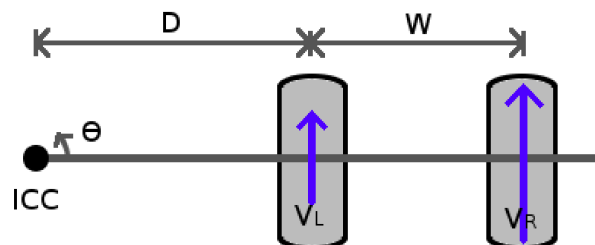


Figure 2.3: Components of differential drive system.

The motion model for a differential drive system is based on smooth motion around the ICC and constant wheel arrangements.

## 2.1 Differential drive

The derivation of the components in figure 2.3 can be achieved through initial calibration measurements, the wheel velocities and by using the following formulas:

$$D = W \cdot V_L / (V_R - V_L) \quad (1)$$

$$\Theta = (V_R - V_L) / W \quad (2)$$

Where  $D$  is the distance between the instantaneous centre of curvature,  $W$  is the distance between the wheels,  $V_L$  and  $V_R$  are the velocities of the left and right wheels respectively, and  $\Theta$  being the rotational angle of the motion.

One of the limitations of the differential drive system is the inability to move in the direction along the rotational axes of the wheels. Attempting to move in this direction requires a combination of rotation and translation to occur. One increasingly popular strategy to overcome this limitation is the use of omnidirectional wheels, which can rotate in two perpendicular axes while still allowing the wheel to be driven by a motor. This hybrid between a caster wheel and a driving wheel can be arranged in such a way to allow the sum of the motion vectors between the multiple wheels to direct the motion of the robot (Feng et al., 1989; Voo, 2000). A typical configuration involves three omnidirectional wheels arranged in an equilateral triangle. One of the downside to this is that the freely moving characteristic can lead to drifting, as most omnidirectional wheels do not include breaks to stop the robot.

## 2.2 Synchronous drive and steering

To overcome the drifting issue experienced by omnidirectional wheels while providing the same manoeuvrability, an ordinary wheel can be used in conjunction with another motor to control the orientation of the wheel with respect to the robot base. This allows the robot to move in any desired direction without the need to rotate the robot body. Typically, these systems make use of multiple wheels, much like the omnidirectional wheel arrangement, that are constrained in orientation with each other. This is mostly done for balance, but it can also allow arc motions to occur by using different power outputs for each of the motors spinning the wheels (Borenstein, 1995).

By combining the ideas of controllable and fixed orientation wheels, more complex manoeuvring systems can be developed such as those commonly seen on tricycles and auto mobiles. The center of curvature of these systems lie along the axis of the fix oriented wheels while the distance to the center of curvature from the robot can be controlled by the angle of the steering wheel. For a single steering wheel system, such as on a tricycle, the only parameter controlling the location of the pivot is the orientation of the one wheel. However, for a dual steered wheel system, such as the Ackermann or kingpin steering systems implemented on auto mobiles, the rotation of the two wheels must be proportionally controlled so that all the rotational axes intersect at one point. That said, it is possible to design a system which relies on large amount of wheel slippage for rotation. However, the motions of such systems are very difficult to anticipate due to irregular slippage, as well as causing large amount of wearing to the tyres, thus are avoided in most cases.

## 2.3 Joint based motion

### 2.3 Joint based motion

As advancements are made to feedback sensors, the trend from the traditional wheel approach to a more flexible pedal based locomotion systems have started to appear. This biologically motivated design allows much greater flexibility in the environment it can operate under, but is hindered by a more complex motion patterns that are both self induced and influenced by the terrain (Quinn et al., 2001). The system also incurs a much higher material cost from the complex integration of motors and joints to make up a leg. The software processing costs also increases when accounting for the complex motion and terrain, thus most implementation of pedal robots are still in their experimental stage. An implementation that resolves some of the balancing issue is one that makes use of more than three legs to constantly maintain balance while allowing other legs to move around and position itself for the desired motion.

While the legs provide a flexible means to manoeuvre, a very similar technique can be applied to an arm mounted on the robot to interact with the environment. The kinematics are quite similar the the legs, but focuses on the exact alignment of all of the components, as the shape of the arm is more important than gaits and balance issues.

## 2.4 Traversal mode

There are two distinct models to be used when commanding the robot to move; a velocity based command and a trajectory based command. Although some robots make use of both types in conjunction with each other, dynamically exploring robots often make use of one or the other based on the type of environment and the flexibility of the robot's motion for the given task. A velocity based command is typically used when manually controlling the robot, or where the robot's task is to navigate around without any predefined path. This allows for a simple reactive system where the robot must explicitly poll for new information which will modify the current motion (Garnier et al., 1995; Paromtchik & Nassal, 1995; Rives et al., 1993; Ward & Zelinsky, 1997). Although this approach does not need to spend time planing the path, it can often lead to unnecessary motions or not detecting various targets if the polling is not conducted at the appropriate interval. The trajectory based command allows for the robot to explore the world more consistently and accurately, but does not adapt well to changes in the environment as the path of traversal is required before the command is given. This usually results in slower motion, but can ease the processing load by assuming the planned path is free of obstacles.

## 2.5 Current configuration

Based on the simplicity and the typical operating environment for the mobile robot, a differential drive system has been used in the current implementation. Table 2.1 summarises the characteristics of the various steering systems discussed above.

A belt drive system is used to connect the wheels to the motors, such that the wheel positions could be carefully adjusted to be directly positioned in the middle of the robot. The wheels measure at 49.5 mm in radius, while the foam tyres are 11 mm



## 2.5 Current configuration

in thickness. The foam was placed on the wheels to promote smooth motion on most surface types as it can absorb most of the small bumps on the surface. The width of the wheels measures at 39 mm across at the ground contact point, while the length of the ground contact area was measured to be 31 mm on a solid surface and 36.5 mm on a softer surface, such as carpet flooring. This difference is due to the softness of the surface which allowed the castor wheels to sink into the ground. There are three castor wheels currently present that operate on a roller ball principle. These wheels control the minimum elevation, which measures at 16.5 mm on a solid surface, while it can reach around 11 mm on soft surfaces. The elevation usually plays an important role for outdoor robots where small obstacles that can scrape and damage the bottom of the robot must be avoided. This is not as important for indoor robots, since the surface tends to be flat with very few exceptions such as small steps between different surfaces, frames on walkways and cables running on the floor. The wheels were placed near the outer edge of the robot, at just 33 mm inwards from the edge, to allow greater control over rotation, as well as increasing the overall balance of the robot to avoid rocking motions. Figure 2.4 show a sample blueprint for the robot base, while figure 2.5 shows the dimensions.

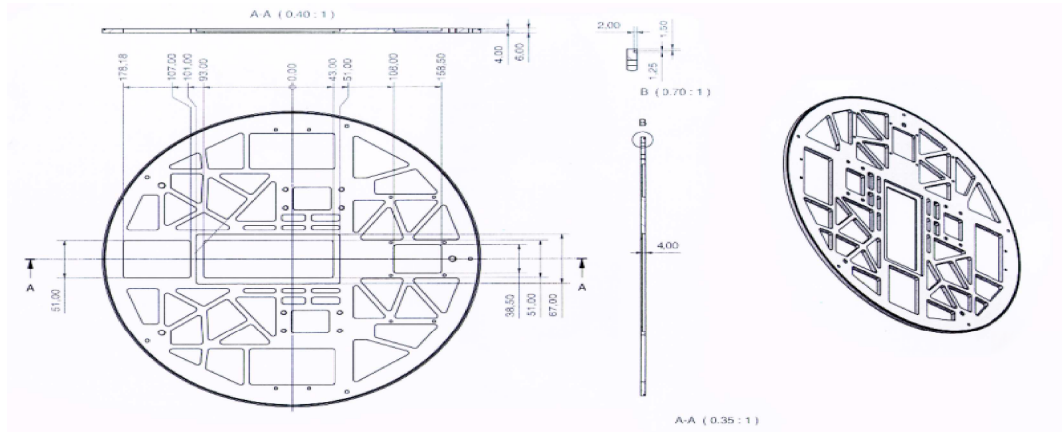


Figure 2.4: Sample blueprint for the robot base.

The robot was built from scratch to allow flexible adjustments and expansion.

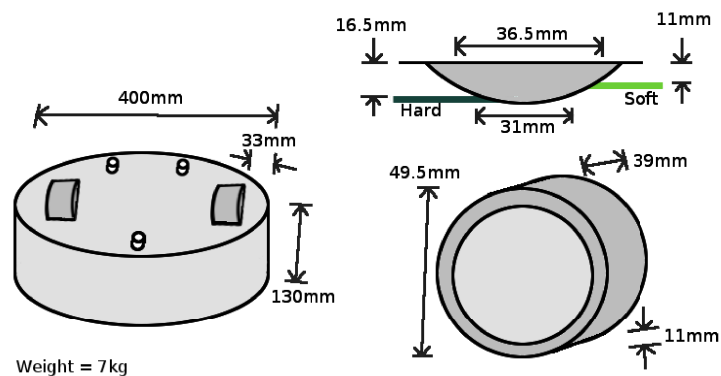


Figure 2.5: Dimensions of the robot base.

The left image shows the bottom of the robot, the top right shows the wheel in contact with the ground, and the bottom right shows the dimensions of the actual wheel.

## 2.5 Current configuration

Table 2.1: Characteristics of common steering systems

Name	Degrees of freedom	Typical number of motors	Motion model	Common issues
Differential	$y, \Theta$	2	Simple	Caster wheels can introduce motion model errors.
Synchronous	$x, y, \Theta$	2 x 3	Simple	Not suitable on rough surfaces.
Ackermann	$y$	2	Simple	Not suitable in confined spaces.
Pedal	$y$	2	Complex	The motion of the body is very rough.

The compression of the wheels due to the weight, cushioning of the tyres, and the ground meant that the exact parameter values for forward kinematics are very difficult to achieve, especially on soft surfaces. During the calibration phases, the ratio of motor axial rotation and the traversed distances were measured to approximate the robot motion on a soft carpet floor so that it can be used as a rough measure for traversing. This configuration allows a very simple base for an indoor mobile robot with opportunities to develop further modules to make improvements later on. The base can also be replaced in future implementation if other locomotive configurations are desired.

Since the primary objective of the robot is to explore the environment autonomously, the motion model has been set to the trajectory based approach, that is, the command is sent as a distance measure. This allows for a sophisticated path planning algorithm to be implemented. The velocity and acceleration parameters being loaded to the motor is currently fixed when exploring autonomously, but can be modified quite easily to allow variable speed navigation in case the robot needs to travel to a destination quickly. The commands, however, are still kept as distance measures, so the overall motion of the robot will be a cycle of stop-and-go operations.

## Chapter 3 – Sensors

In order for the robot to successfully operate within the environment, it must sense and acknowledge the surroundings, as well as possessing the ability to sense the states of itself. Being able to receive and utilize these input information allows greater range of applications for the robot, as it allows adaptive behaviours to occur. The major drawback of closed-loop systems against open loop systems is in the complexity of interpreting and integrating the sensor readings to the internal state with minimal errors such that the sensors can make use of the updated states about the environment and itself to enhance the subsequent calculations.

There is a wide range of sensors that are available for registering many different types of signals, but unlike simulated environments, real world sensors are influenced by environmental conditions, limitations in the sensor range or capacity, and the characteristics of the device itself, such as the capture time and sensitivity to noise (Betke & Gurvits, 1997). In addition to these considerations above, other issues that are often ignored in simulations include the material costs, power consumptions, physical placement constraints, and the general applicability for the particular environment.

The sensors mounted on the mobile robot can be broadly classified into two categories in terms of what it senses; internal sensors for measuring the attributes about itself, and external sensors which interacts with other systems, like the environment, to obtain some information about them or in turn, about the robot. In the case of external sensors, there is a further classification which can be made on whether the sensor makes use of the ambient signals, called passive sensors, or emits signals to the environment to be detected when it returns to the sensor, called active sensors. Both types of sensors have their own benefits, such as low energy consumption and non-intrusiveness behaviour for passive sensors, while low ambiguity and a wider range of operating environments are some of the benefits in using active sensors.

### 3.1 Internal sensors

Internal sensors are responsible for measuring attributes of the robot itself, thus play an important role in grounding the other sensor readings by specifying the state of the robot. These sensors are often not as interesting as the external sensors due to the controlled manner in which they operate since they lack the interactions with other systems.

#### 3.1.1 Timer

The most fundamental of all signals feeding in to the robot is the timing information from a clock. Whether it is to simply time-stamp particular events or to perform complex synchronisation tasks, the clock forms the foundation for ordering

### 3.1.1 Timer

the events in some consecutive sequence (Lamport, 1978). Most systems often rely on a single central clock to coordinate the events, but extra considerations must be made when combining multiple clocks, such as offsets and drifts, that can be determined through a calibration phase or through continuous synchronisation in case of irregular drifts during operation (Lamport et al., 1982; Mattern, 1989; Mattern et al., 1991; Schwarz & Mattern, 1994).

### 3.1.2 Rotary encoder

Another commonly used sensor can be found on the motors for measuring the shaft positions using an encoder. This position indicator is bundled on most commercial motors and can be used to determine the degree of rotation of the motor shaft from some given position. Using this value, it is possible to anticipate the motion behaviour or estimate the required motion for the attached components if attributes such as the circumference and gear ratios are known. The accuracy and reliability of these encoders are quite reasonable, except for rare mechanical failures from wearing or excessive stress on the rotating shaft. However, the reliability of the system which makes use of the encoder values can vary significantly on the characteristics of the attached components and environment they operate in, such as the quality of the wheels.

### 3.1.3 Power indicator

A measure that is commonly seen in commercial products is the power indicator based on the remaining battery charge. This information is often not considered in many researches based robots due to the irrelevance to the mechanical or algorithmic development. The lab environment also provides simple means to recharge the battery or the robot does not consume a large enough power in a single execution. For researches that involve specific power outputs, such as being able to determine the precise power output of motors, the information allows algorithms to consider modifying other attributes to counterbalance the effect of the reduced battery or to simply warn the user (Martin & Seiwiorek, 1996; Oh et al., 2000).

## 3.2 External sensors

While the use of internal sensors provide useful information about the state of the robot, the external sensors involve a much more complex process to cater for the broad range of information through interactions with the environment. The generic tasks for these include the validation and translation of the sensor signals given the current environmental conditions. The sensors are often categorised by the modality of the detectable signals, but they can also be categorised by the type of information it generates. Considerations in selecting the external sensors includes the usability with respect to the characteristics of a typical operating environment, reliability, precision, and the material cost of the device, which can range significantly between high and low quality implementation of the same type.

### 3.2.1 Passive sensors

### 3.2.1 Passive sensors

Often referred to as inertial sensors, accelerometers and gyroscopes are sometimes fitted to mobile robots that manoeuvre on different altitudes caused by non-flat terrains, bumpy motions, as well as the tilting of the robot body. These sensors play an important role in tracking the 3D pose changes and maintaining the robot's balance. These sensors can also play a role in 2D environments by monitoring the change the robot's state to support the other sensor readings.

A similar sensor to the accelerometer is the inclinometer, which is used to determine the current pitch and roll of the robot using the acceleration provided by gravity. Due to the common occurrence of drifts in some sensors, the inclinometer is commonly used to ground the measurements to avoid potential hazards from incorrectly measuring the balance. Similarly, instead of accumulating the yaw changes, absolute values of the strength of the Earth's magnetic fields can be measured on compass sensors to identify its current orientation (Duckett & Nehmzow, 1998). These sensors are prone to electrical noise, thus must be mounted on an extended arm to stop the metal body and the electronics from influencing the signal.

### 3.2.2 Ranging sensors

A more commonly applied sensor in a research environment used on mobile platforms is the ranging finding sensor. Several different methods and modalities are used in identifying the region of open space between the sensor and an obstacle. These are often used to detect the presence of an obstacle in the immediate vicinity that is visible to the particular sensor, but can also be used to identify multiple obstacles as well as the attributes of the objects by exploiting the characteristics of the type of signal being used.

#### 3.2.2.1 Sonar

The sound navigation and ranging (sonar) sensor has long been a popular addition to mobile robots due to its low cost and simplicity of operation. A sonar sensor typically operates by detecting the time of flight or the phase shift of the high frequency chirp they emit, which is detected after it bounces back to the receiver. Although its behaviour appears simple, the sensor is affected by many other contributors, such as the speed of the chirp in the medium, interference from other noise sources, operational range, angle of incidence, absorption and reflectivity on surfaces, echoes, and a cone shaped dispersion of the sound signal (Drumheller, 1987; Dudek et al., 1992; Kleeman, 1999). Figure 3.1 shows sample signal strength of a sonar chirp at varying angles from the emitter. Note that the time of flight is unaffected by the loss in the strength, but the smaller lobes can potentially become an unwanted signal if the threshold is not appropriately set. Some of these issues can be corrected trivially through the use of multiple sonar sensors or by combining the reading with other types of sensors. There currently exists a wide range of research dedicated towards making better use of the sonar sensors in the attempt to achieve high precision object detection and map construction systems such as those found in biological systems (Araujo & Grupen, 1997; Bank, 2002; Borenstein & Koren, 1995;

### 3.2.2.1 Sonar

Borghini & Tosolini, 2007; Cahut et al., 1998; Chong & Kleeman, 1999; Crowley, 1989; Goel & Sukhatme, 2000; Kleeman, 2003; Kleeman & Kuc, 1995; Varveropoulos, 2000; Wijk, 2001).

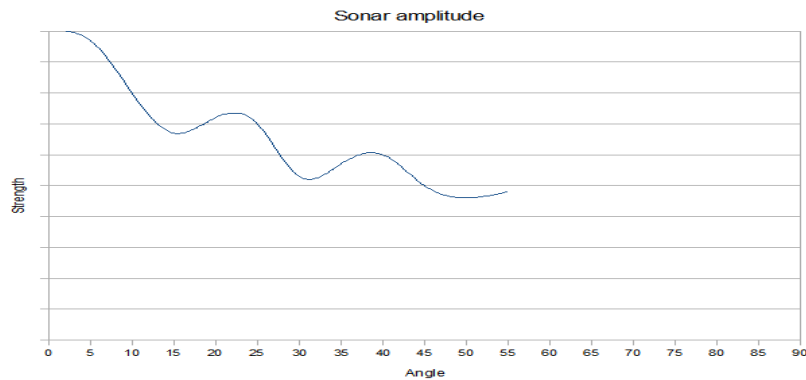


Figure 3.1: Typical sonar signal strength at varying angles. The lobes of the sonar sensor signals can result in detecting secondary reflections if the appropriate thresholds are not considered.

### 3.2.2.2 Radar

While the sonar sensor operates using an acoustic signal, the radio detecting and ranging (radar) sensor operates by using the phase and frequency characteristics of radio waves within the electro-magnetic spectrum to measure the distance to an obstacle. The sensor has similar behaviours to that of the sonar sensor, both in the way it operates and the problems they face, with the exception of the material cost, the ability to operate without the need of a dense medium to traverse through, the speed, and the ability to penetrate certain surfaces which can provide useful information about objects that may not be directly in sight (Bahl & Padmanabhan, 2000; Clark & Whyte, 1997; Foessel-Bunting, 2000).

### 3.2.2.3 IR

Another type of electro-magnetic wave based sensor which has similar characteristics to the above is the infra-red (IR) range finder. This type of sensor uses the reflected light's intensity or a triangulation process to determine the distance to the obstacle. Due to the wide variety in the reflectivity of different surfaces and the scattering of light from Lambertian surfaces (Oren & Nayar, 1995; Poulin & Fournier, 1990), the signals can sometimes be inconsistent. The low cost device is often used as a non-contact bumper for obstacle avoidance to protect sensitive equipments (Borenstein, 1989; Kwon & Lee, 1995).

### 3.2.2.4 LIDAR

Light detection and ranging (LIDAR) is regarded as the most accurate and reliable sensor for measuring distances to an obstacle, as the devices emit a very narrow beam to a well controlled direction. The operational range of these sensors can

### 3.2.2.4 LIDAR

extend from a few meters for low powered beams, to thousands of kilometers for some of the high end sensors used in fields such as astronomy. Although the high precision is desirable, the sensor comes at a very high cost, which is often a key deciding factor in research projects. They are also potentially more hazardous than the sensors above due to the focused beam which can enter the human eye when operating in populated areas, especially in indoor applications. Another potential drawback is the reduced viewing area which can affect the usability in certain tasks which require a broad coverage of the area instead of a precise measure of a single point. To overcome this, some implementations make use of other sensors in conjunction with the laser range finders (Aboshosha & Zell, 2003; Castellanos et al., 2001; Diosi & Kleeman, 2004; Dudek et al., 1996; Elgazzar et al., 1997; Kelly, 1994; Lewis & Johnston, 1977; Lu & Milios, 1997; Neira et al., 1999).

### 3.2.2.5 Bump and tactile sensors

While the range finders mentioned above are all examples of non-contact sensors, there are also a range of contact based sensors available to detect degrees of contact with the robot. Often referred to as a tactile sensor, it involves a switch or a pressure sensor to determine the degree of potential energy involved in the collision. Depending on the system, this type of sensor can vary in purpose from simple obstacle detection to determining the appropriate forces required to grip onto something with a robotic arm. The range in sensitivity, as well as the amount of sensor points determines the cost involved in implementing the tactile sensor, but the use is typically limited to systems which requires sensitive pressure information or a very cheap implementation of a collision detector (Krotkov, 1991).

### 3.2.3 Camera

With the price of cameras becoming more affordable, the improvements in the quality, and the development of sophisticated image processing algorithms, the popularity of visual sensors have dramatically increased in recent times (Bigas et al., 2005; Smith & Brady, 1997). There exist vast repertoire of approaches in extracting useful information by analysing the pictorial data that is based on the intensity values of the reflected electro-magnetic waves and the position of the detecting photo-sensor. These techniques can range from simple colour detection to a more complex feature tracking algorithms using spatial and temporal relationships (Whelan & Molloy, 2001). The major differences between the camera sensor and the other light based sensors discussed above are in the passive modality, where it makes use of the ambient light rather than emitting any controlled beam for the sensors to operate, the large number of simultaneous measurements that can be made from the array of photo-sensors within, and also the directional information that are extracted instead of the distance information.

The increased number of simultaneous measurements is achieved through the array of individual photo-sensors that are present within the single device. This characteristic allows for the camera's greatest benefit, which is the ability to capture the scene structure by taking a snapshot of the inter-pixel intensity information in a single instant. This ability allows the camera sensor to identify and track objects within the scene with ease while providing more information with regards to the

### 3.2.3 Camera

relationships between neighbouring views such as the presence of shadows, surface structure from lighting changes, textures and patterns for segmentation, and when combined with other sensors, it too can determine the distance to obstacles (Murray & Little, 2000; Narkhede & Golshani, 2004; Zitnick & Kanade, 2000).

The availability of more data gives rise to more sophisticated algorithms with aims of one day outperforming the abilities of those in biological systems (Navalpakkam & Itti, 2005; Soyer et al., 2003). This development requires both hardware and software improvements, as the digitised approximations become detailed enough to distinguish the most subtle directional and intensity information, while the processing capability increases to allow more information to be analysed.

The most common visual sensor makes use of the visible spectrum as it best mimics the human visual system, as well as their low cost and availability. However, there exist other sensors that are designed to capture the reflected light in the other parts of the electro-magnetic spectrum. These types of sensors are often used in specialised tasks, such as observing the ambient thermal radiation of an object or to avoid interference from the visible spectrum.

### 3.2.4 GPS

The global positioning system (GPS), which was originally intended to be used for stealth localisation by the military, has become a popular source of latitude, longitude, altitude and time information which are delivered from multiple satellites orbiting the Earth. Its usage has typically been focused on outdoor systems due to the direct line of sight and the constant density of the medium of travel that are required. The satellites transmit their current coordinate points which the system uses to triangulate the current pose. Due to the wide coverage using a small number of satellites, the accuracy that can be achieved is not as high as required for robots operating in confined environments, but can be combined with other sensors to derive a multi-scaled pose (Li & Hayashi, 1998).

## 3.3 Sensor configuration

As extension modules are developed and integrated with the mobile robot, extra sensors can be placed to observe the environment using different modalities and approaches. In the current implementation, the robot consists of six different modules; the base, IR sensor array module, sonar module, floor tracking module, directional camera module, and an omnidirectional camera module (Baker & Nayar, 1998; Spacek, 2005). The base, as described earlier consists of a cylindrical frame with space inside for the controller boards, battery, the motors with their encoders, and the wheels. The IR sensor array is arranged on a disc which is placed on top of the base. The sonar module currently sits on top of the IR sensor array at the front of the robot, while the controller boards for the IR and sonar modules sit towards the back of the robot on top of the IR array. The controller board also contains the communication ports, which can be used to relay commands to another computer. Shielding the controller boards is the laptop mount to allow high level processing in place of the microprocessors. To the sides of the robot are the floor pointing mounts and cameras, while the directional camera is currently placed on top of the sonar



### 3.3 Sensor configuration

module to provide the panning motion by reusing the servo motor used on the sonar sensor. In future implementations, the directional camera will be given its dedicated servo motors for yaw and pitch control. Lastly, the omnidirectional camera is mounted on top of the laptop mount to minimise viewing itself as it observes the surrounding environment.

#### 3.3.1 Encoder

Using the rotational encoders on each of the motors, the ratio between the pulse counts and the distance the robot has traversed was determined. This simplified the complex wheel compression and gearing factors by assuming constant motion characteristics during operation. This ratio was then used for forward and inverse kinematic calculations to predict the motions and plan the paths of traversal. This simple mapping allows the high level programs to deal in a more consistent unit rather than having to know the exact pulse requirement to use the device. It is important to note that this open-loop approach is only an approximation and the calibration value is never precise or consistent across different environments. With the existing set-up, the ratio was determined to be 801603 pulses to one meter when tested on a solid surface with no visible signs of slippage. This value was calculated by specifying a certain number of pulses to count to, then measuring the traversed distance. The ratio was then used to verify the correct pulse to distance conversion on the same surface.

#### 3.3.2 Infra-red module

Within the IR sensor module, the sensors are placed in a circular arrangement around the robot to allow simultaneous measurements of the perimeter of the robot. This allows the sensor array to act as a non-contact bump sensor to avoid collision as well as determining the distance to obstacles that are close by. Instead of measuring the change in intensity or shifts, which is heavily dependant on the reflective properties of the obstacle's surface, the current sensor, the Sharp GP2D12 ranger finder as shown in figure 3.2, makes use of a triangulation technique to measure the distance to the obstacle. The sensor is equipped with an emitter and a detector, which contains a linear charge-coupled device array to detect and determine the incidence angle of the reflected light. Due to the known distance between the emitter and the detector, the distance to the obstacle can be derived for a particular angled beam from the emitter. Figure 3.3 below illustrates this process, where  $D$  equals the distance between the emitter and the receiver, and  $\Theta$  represents the angle between the incidence ray and reflection.



Figure 3.2: The Sharp GP2D12 infra-red sensor module. The two lenses are the emitter and the receiver of the IR beam.

### 3.3.2 Infra-red module

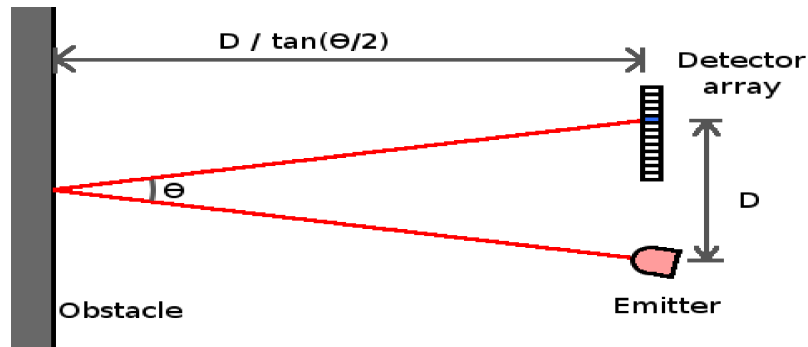


Figure 3.3: Deriving the distance to an obstacle with the IR sensor. Triangulation is used to determine the distance between the IR sensor and the object.

This approach requires the obstacle's surface to be parallel to the sensor's face or the surface must exhibit Lambertian behaviour such that the emitted beam can bounce back into the detector, thus can potentially be problematic in certain environments and orientation as it may miss the obstacles or incorrectly measure the distance. However, this technique in measuring the distance overcomes some issues such as interference from ambient light and the absorption properties of the obstacles which can reduce the intensity and result in significantly erroneous readings.

One of the major downside to this approach is the very limited range of surface orientations that are allowed for the surface materials. With increased distance to the obstacle, the amount of light entering back to the receiver decreases dramatically. A slight misalignment in the surface orientation results in the object not being observed until it is quite close to the robot where the tolerance to the surface orientation is greater. Figure 3.4 shows the sensor readings at various surface orientations when the obstacle is placed at 10 cm intervals away from the robot. The shift in the peak is due to the alignment of the emitter and receiver, as well as the non-grainy surface, which preferred the surface to be tilted slightly towards the receiver.

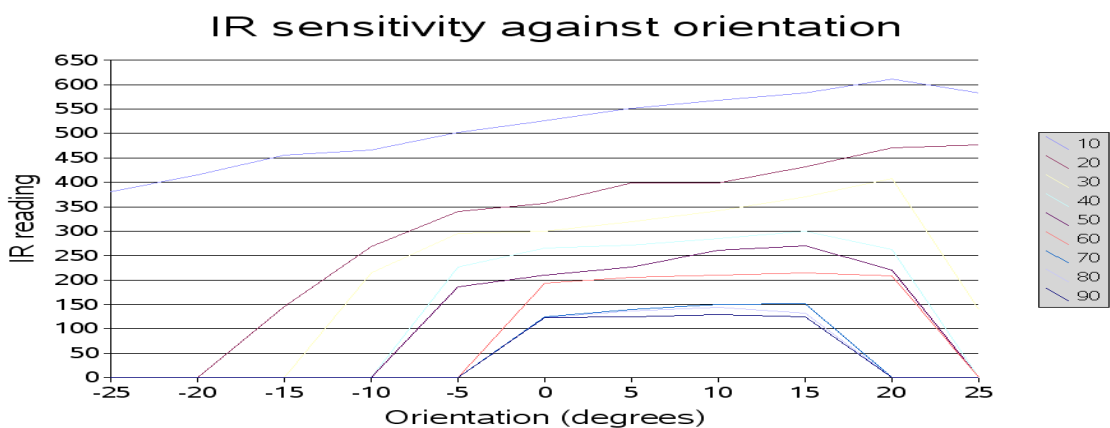


Figure 3.4: IR sensor sensitivity against various surface orientations and distances to the surface.

There is a steep sloping of the sensor reading as the orientation is changed. The peak is not centered due to the arrangement of the emitter and receiver. The lines represent the distance between the sensor and the surface.

### 3.3.2.1 Placement

### 3.3.2.1 Placement

The module is designed with a capacity of twelve IR sensors arranged in a ring, equally spaced at 30 degrees apart, which can be seen in figure 3.5. Due to budgeting reason, the current system only uses eight, which are placed at the 30, 60, 90, 120, 150, 210, 270, and 330 degrees. The arrangement is designed to anticipate most of the obstacles to appear as the robot from the front, while still allowing some detection of obstacles behind itself for dynamic objects and when it has to reverse. Note that cross-talks between the sensors do not occur due to the large angle of separation between the sensors.

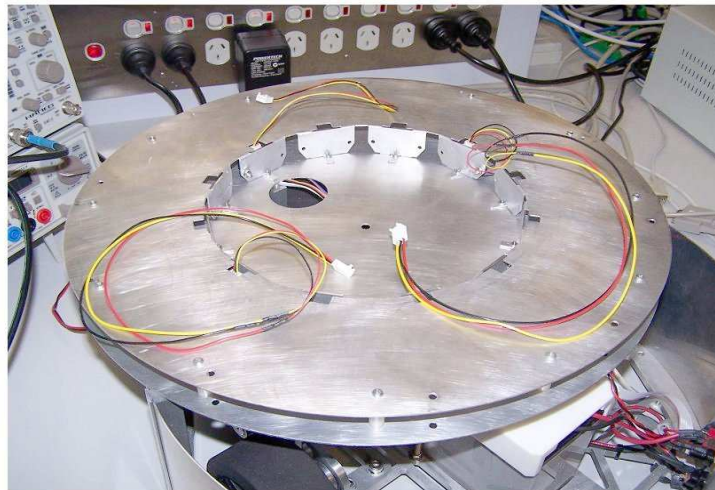


Figure 3.5: Infra-red sensor mount.

The photo shows three IR sensors that are mounted on holders spaced 30 degrees apart and 10 cm in from the outer edge of the robot to avoid overlap and ambiguous sensor readings when the obstacle is too close.

The sensors are embedded inside the robot by 100 mm with the operational range of the sensors in mind. The minimum distance of operation was suggested by the hardware specification manual, but is actually the distance to the back plate to which the sensors are mounted on. The actual distance from the sensor to the outer edge of the robot is measures at 85mm, which allows for a slightly larger operational range. This also means that the sensors can potentially give false readings when the obstacle is almost touching the robot, but since this is not a safe operating range for the robot, obstacles should never get this close to cause any ambiguity.

### 3.3.2.2 Sensor reading

Most active range finders have limited ranges they can operate in, whether they are due to the hardware or the modal characteristics. In this particular case, the limitation is governed by the sensitivity of the charge-coupled device arrays and the reduction of the signal strength over large distances. The hardware specification state the operational range to be from 100 mm to approximately 800 mm with a beam width of approximately up to 160mm, but these values varies from sensor to sensor, thus a calibration process was carried out on each of the sensors.

### 3.3.2.2 Sensor reading

Using a rough grained black plastic surface, distance measurements were taken for each of the sensors while the obstacle was placed at controlled distances away from the sensors. The average readings for the sensors are summarised in figure 3.6, where the deviation tended to remain low and consistent for various distances at around 5 to 10 units as the surface orientation was kept parallel to the face of the IR sensor. The surface material was chosen to find the smallest upper bound on the distance, due to the scattering and the loss of intensity, thus allowing a reasonably safe assumption that the obstacle is detectable at a certain distance away before the signal is lost.

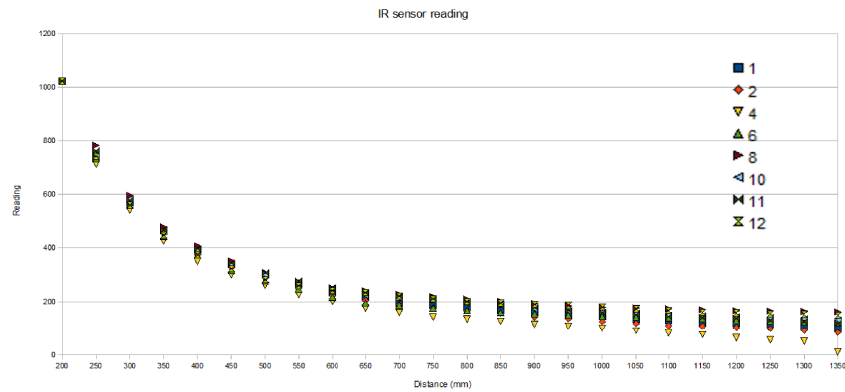


Figure 3.6: Distance measure against each IR sensor's readings. The relationship is slightly different between different sensors, thus require individual calibration.

The distance measures used for figure 3.6 are taken from the center of the robot to the obstacle to ease the mapping required to convert between each sensor's coordinate system to the one of the robot at a later stage. The radius of the robot is 200mm, thus anything below that are considered to be inside the robot which cannot occur. It was found that the saturation point of the sensor was not a single point, but covered a small range of distances due possibly to the sensor hardware limitations or the characteristics of the surface. Variation between the different surfaces when the obstacle was placed at a fixed distance showed a wide range of fluctuation, as shown in table 3.1, indicating the necessity for supporting the measurement by using alternate sensors.

As the trend indicates, the behaviour after the 200 mm mark shows an inverse power like behaviour. Instead of using a complex polynomial approximation to represent this curve, it is also possible to convert the curve into series of linear transitions between the calibrated points, especially with a small enough interval. This involves iterating the array of calibration values until the appropriate interval is found, then interpolating the two adjacent values to find the proportional distance measure for the given sensor reading. Although there is the cost involved in iterating and interpolating between the two calibrated values, this process is reasonably fast for small number of calibration points and can often approximates the curve better or faster than using a single function which models the whole curve quickly or accurately. To speed up the process, it is also possible to consider algorithms such as a binary search algorithm or by remembering the previous interval to start the search due to the continuity in the distance measures between consecutive data readings.

### 3.3.2.2 Sensor reading

Table 3.1: IR sensor reading at 50 cm for various materials

Material	IR reading
Plastic	181
Matte paper	184
Glossy paper	190
Silk	198
Cotton	202
Carpet	191
Rubber	190

The pseudo-code for mapping the sensor readings to distance measures using the linear interpolation approach and remembering the previous value is shown below in algorithm 3.1.

```
set index = len(calibration_array) - 1
function IRSensorValueToDistance(value):
  while true:
    set upper = calibration_data[index]
    if value <= upper:
      set lower to calibration_data[index + 1]
      if value > lower:
        set weight = (upper - value) / (upper - lower)
        return minimum_distance + interval * (index +
          weight)
      else
        index++
    else
      index--
```

Algorithm 3.1: Linear interpolation mapping from sensor value to distance for the IR sensors

The calibration phase also involved the measurement of the dispersion angle for the emitted beam. Due to the spreading of the beam from the emitter and the scattering which occurs at the obstacle's surface, the sensors could detect the obstacles that existed in a cone shaped viewing area. Since the distance to the object is measured, this resulted in an arc shaped positions for the obstacle. The approximation to a cone shape is not entirely accurate due to other factors such as the reflectivity and sensitivity at different incidence angles, but is a reasonable approximation of the behaviour of the scanned area by the sensor. By using the locations of sharp changes in the sensor readings as an obstacle was moved in an arc across the front area of the sensors, the viewing angle of the sensor was determined. Since the variation on the viewing angle for the different sensors were not significantly different, at less than 1.5 degrees, they were approximated to the smallest of the measured viewing angle of 10.5 degrees to underestimate the vacancy to avoid potential collisions.

### 3.3.2.3 Motion limit

#### 3.3.2.3 Motion limit

As small obstacles that are very close to the robot lie in the blind spots of most sensors, it is crucial for the robot to avoid allowing anything to move too close. To provide the robot with a safe buffer zone, a motion limit operation has been included at the controller board level, which overrides the current motor command with a stop command if an obstacle is detected within approximately 150 mm from the outer edge of the robot. To allow the robot to continue its operation safely, the motion limit sets a flag to limit the direction of the subsequent move command. To simplify the direction of the traversal, if the IR sensors located at the front of the robot triggers the override, the robot then only accepts a backward motion command or rotations and the reverse is done for the back sensors. Initially, the robot was allowed to perform a rotation at any time, but due to the addition of extra modules that extended beyond the circular boundary of the robot, a rotation meant it would cause portions of the robot to collide with obstacles. To prevent this from happening, the sensors located at the 30 and 210 degrees positions are used to stop left turns, while the sensors at the 150 and 330 degrees positions were used to limit right turns. These sensors were chosen as they can view the adjacent space to the floor viewing cameras, which extend out to the side of the robot by approximately 10 cm. Figure 3.7 below illustrates the allowed motions after the sensor has detected an obstacle within the motion limit area.

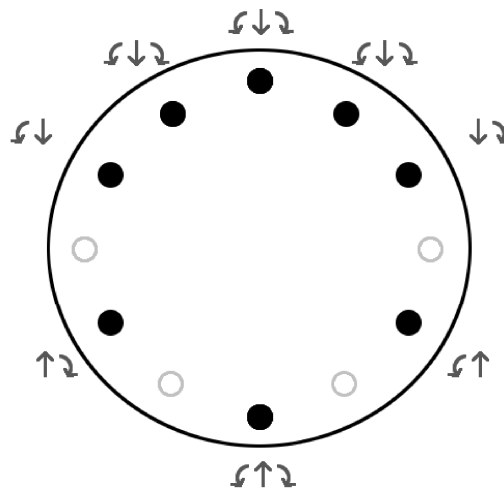


Figure 3.7: Available motion directions after motion limit. The black dots represent the location of the IR sensors, while the white circles are currently unoccupied.

This approach allows the robot to keep operating automatically while avoiding collisions with the environment. A more traditional emergency stop mechanism can also be included in tele-operation scenarios.

### 3.3.3 Sonar

Some of the major limitations of the IR sensor include the large variety of reflective surfaces and the short operating range that is available for the device. The

### 3.3.3 Sonar

sensor often reports varying distance measures, which can fluctuate quite significantly due to the orientation, especially at longer distances to the object. By emitting acoustic chirps, the sonar sensors are able to receive a more consistent distance measure as the surface types does not affect the time of flight. Although the measurements are still dependant on the behaviour of sound in the environment, such as the temperature, humidity and air density, the readings typically stay consistent with minimal variation during a single experimental run. The sonar sensors that are used, the Polaroid 6500 ranging modules (Polaroid, 1995; SensComp, 2004), operate at a frequency of approximately 49 kilohertz, which is outside the range of most ambient noise.

#### 3.3.3.1 Placement

Due to the low cost and reasonably consistent performance, there are large range of research that incorporate the sonar sensors. These include the investigation of the acoustic behaviours to sophisticated object localisation techniques by combining multiple sonar readings. The current module consists of two sonar sensors mounted at the front of the robot that are spaced 156 mm apart and oriented in the same direction as shown in figure 3.8. This stereo set-up allows simple error correction between the two to identify disparity in the object's surface angle and erroneous readings from echoes and dispersed signals.



Figure 3.8: Sonar modules on top of the servo motor. The sonar sensors can be rotated independently of the robot's orientation, thus allowing greater flexibility in the sensor usage.

Since the two sonar sensors overlap in their area of coverage, the sensors face the problem of cross-talk between each other. A simple solution has been implemented to compensate for this behaviour by setting one of the sensor to be a dedicated receiver, while the other sensor switches between the send and receive mode (Vilmanis, 2005). This allows both sensors to use the same chirp to identify the difference between the times of flight to identify the location of the obstacle. The simple triangulation technique is illustrated below in figure 3.9, where TOF is the time of flight.



### 3.3.3.1 Placement

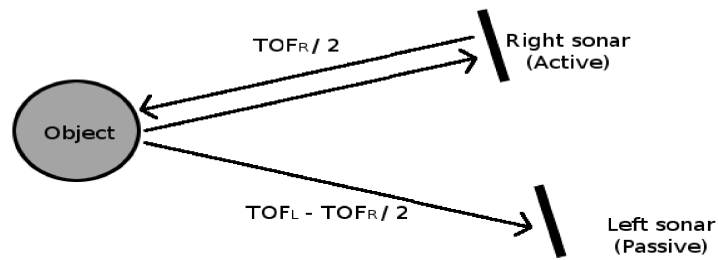


Figure 3.9: Finding a target using a dual sonar behaviour.  
Only one sends the chirp to avoid cross-talk between the sensors.

### 3.3.3.2 Operational range

To allow the scanning of different areas without having to rotate the robot, which can be imprecise due to the interactions with an unknown surface, the sonar sensors are mounted on a servo motor to control the orientation of the sending and receiving faces. The servo motor which was used, the Hitec HS-422 Deluxe, is set with an rotational range of  $-60$  to  $+60$  degrees with a single degree of granularity. The higher level of precision it is capable of does not provide a great deal of extra benefit, as the width of the sonar chirp is not consistent, nor can be accurately measured. The maximum and minimum range is set to stop the sonar sensor from seeing the other components of the robot located behind itself and to avoid the wires getting tangled.

The operational range of the sonar sensors are controlled by the hardware based limitation on the time taken to switch between the sending and receiving modes and the loss of signal strength when the chirp is sent over a long distance. The sonar sensor must account for the residual ringing within the module after sending the chirp, called the blanking time, which is currently set to be approximately 2.4 milliseconds. This value can be modified by sending an override bit, but since this value also contributes to the minimum time of flight, the distances should account for the ideal distance of the obstacle. To achieve a more reliable measurement, the two sonar sensors should operate in a similar manner. This means the object should be viewable by both sensors, thus the ideal distance starts from when the viewing area of the sensors start to overlap.

The maximum operational range for the sonar module, according to the hardware specifications, is at just over 10.5m. However, in an indoor environment with large number of obstacles, it is unlikely that a sensor measurement from such a long distance will not be corrupted by noise such as echoes and secondary reflections. By setting the sonar module to anticipate for long distances, it also means waiting for the time-out to occur will greatly reduce the sampling speed of the sensor. Although the detection of obstacles that are far away may not be reliable, the availability of this information allows the module to be deployed in other environments, as well as giving a rough indicator of vacancy to be used for path planning and can avoid frequent updating of paths. The module is currently set to time out after approximately 30.84 milliseconds, which translates to roughly 5m, as the chirp is required to traverse back to the receivers.



### 3.3.3.2 Operational range

Much like the IR sensor, the sonar sensor also suffers from the dispersion of the signal after it is emitted from the device and reflects off of the obstacle's surface. The behaviour of the acoustic signal is slightly more complicated than the light sensors due to more ambiguity from secondary waves, multiple reflections from the increased range, and the wider wave front. A detailed investigation into the applicability of sonar characteristics has not been conducted for this project, but significant attributes such as the dispersion angle for the primary lobe of the sound wave strength has been measured so the sonar's most prominent signal can be determined. To measure this angle, a similar process to the IR sensor was carried out, but using a small grained plastic plate. The rough surface was intended to induce some scattering to make sure the signal reflected back to the sensor. Since the type of material only affects the amplitude of the signal, as long as the reflected signal was strong enough to be detected, the absorbency did not affect the distance measure. The experiment showed that the beam was reasonably narrow at 7.5 degrees in each direction from the perpendicular line to the face of the receiver.

### 3.3.3.3 Sensor reading

The calibration process indicated three distinct regions for the sensor operation; the blanking and blind zone for the passive sensor before approximately 450mm, a linear trend in the middle, and the region after the cut-off point where the sensors time out. The results of the calibration is summarised in figure 3.10. The dividing point between the first two ranges is the minimal distance the sensor can operate due to the single emitter approach, which can be derived from the viewing angle and the distance between the receivers. The distance between the receivers is 120mm, thus giving the theoretical minimum distance of approximately 455.7mm. To account for possible errors, the minimum distance was set to a round value of 500mm. The maximum value was simply set to when the sensors timed out, which then reports an empty reading.

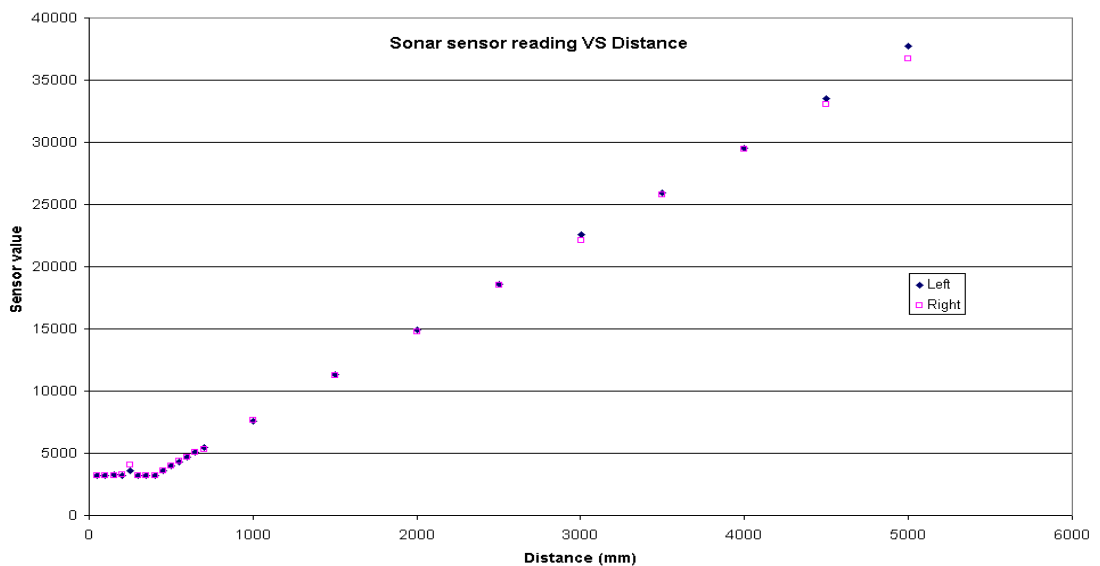


Figure 3.10: Time of flight versus distance measure for the sonar sensors. The linear trend and the operational range for both receivers can be observed.

### 3.3.3.3 Sensor reading

Within the valid zone, the linear trend for each sensor has been modelled into a simple linear function to convert the time of flight into a distance measure. The relationship is not consistent across different environments, but remains reasonably accurate unless significant changes occur in the operating environment, such as the toggling of the air conditioner. Note that the sensor values correspond to the clock counts, which is proportional to the time of flight. Rather than a two step conversion, the sensor value can be converted directly with the following function;

$$\text{right\_distance} = 0.135 * \text{right\_clock\_count} - 26.761 \quad (3)$$

$$\text{left\_distance} = 0.137 * \text{left\_clock\_count} - 43.209 \quad (4)$$

The small intercepts are due to slight inaccuracies in distance measures during the measurement process, but is a negligible amount compared to the magnitude of the clock count. Although the object was placed equally distant from the two sensors, the slight difference in the coefficient indicated that the placement was not perfectly accurate, or the slight delay introduced from the sequencing of the sensor readings. The linear trend showed very consistent results indicating the reliability of the sonar sensors when used in a consistent environment. The consistency between the two readings plays an important role in error detection and when encountering smaller or angled surfaces. This will be explained in further details in chapter 8.

### 3.3.3.4 Tilting

Due to the increase in the likeliness of receiving an erroneous signal when measuring an obstacle that is further away, it can often be advantageous to limit the range of the sensor manually. By reducing the time-out value, it allows faster sensor re-use and reduces the uncertainty in the sensor measurement. However, this approach must consider potentially receiving the first chirp just after the second chirp is fired after it times out. It is possible to cater for this by using multiple frequencies or by introducing a pause while ignoring all measurements that are beyond the threshold distance.

An alternative approach to limit the range of the scan is to modify the direction of the emitted chirp, such that it limits the locations of the surfaces that can reflect the signal back to the receiver. A simple implementation of this technique is to tilt the face of the receiver, such that it points down to avoid the reflected signal from a distant, vertically standing surface. There are also other benefits, such as being able to detect smaller obstacles near ground level and avoiding the detection of obstacles that are too high for the robot. By tilting just the sensors itself, rather than the sonar module, the rotational axis of the servo remains consistent, thus allows for a constant elevation of the sensors to simplify the calculations.

The approach was tested on a carpet and vinyl flooring to note the validity on different surface types. Operation on the carpet floor showed more fluctuation in the viewable angle due to the irregular surface structure, but did not show any significant difference to the overall behaviour. The effects and measurements at various tilting angle for an obstacle placed 2 meters away is summarised in figure 3.11 and 3.12 respectively. The green area shown in figure 3.11 is for a parallel scan to the ground at an elevation of  $H$  and viewing angle of  $\Theta$ , while the blue area shows the tilted version. Note that the actual distance,  $D$ , to a flat surface starts to differ if the sensor

### 3.3.3.4 Tilting

is rotated beyond its viewing angle. The tilting angle,  $\Phi$ , also causes a larger ambiguity in the actual distance to the obstacle. Although most of the obstacles encountered by the sonar are large enough so it does not fall within the newly created blind spots, having multiple blind regions and inconsistencies with other sensors can become difficult to manage. The flat surface also means less of the reflected signal will reach the receiver due to the increase in the angle of incidence against the obstacle. Although these limitations can help narrow the beam to improve its precision, the approach also introduces many constraints on the obstacles it can observe. In the end, the tilting has not implemented in the current set up.

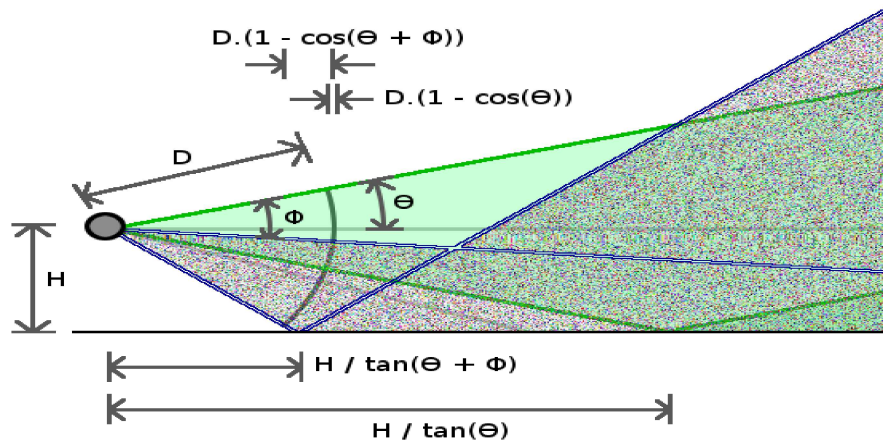


Figure 3.11: Effects of tilting the sonar.

The solid line bound region shows normal orientation, while the spotted region shows the area covered by tilting the sensor by an angle of  $\Phi$ , where the source of the signal is the grey circle.

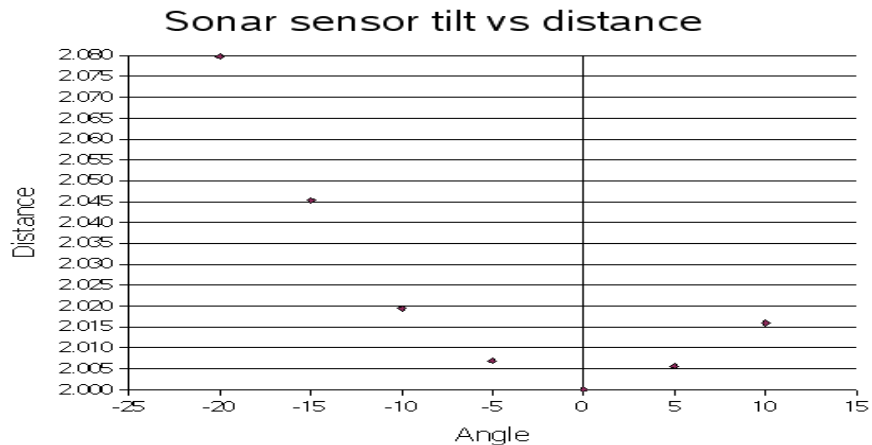


Figure 3.12: Tilting angle versus sensor reading.

The change in the distance to the sonar sensor can be observed by tilting the receiver.

The applications of the IR and sonar sensors will be discussed in more detail in chapter 8 where they are employed for the task of mapping the structure of the environment. The strategies involved in using these range sensors play an important role in constructing a useful map for the mobile robot as the captured data is combined with other sensor readings.

### 3.3.4 Cameras

### 3.3.4 Cameras

The last and the most complex type of sensor included on the mobile robot is the series of off-the-shelf USB webcams with charge-coupled device sensors mounted at different locations on the robot. The cameras are used simultaneously and independently for capturing different aspects of the environment and from different perspectives. Currently, there are four cameras mounted on the mobile robot that make up three different modules. Each module is implemented for specific tasks with some common filter combined with unique parsing algorithms to extract the required information for the task.

As noted earlier, the visual sensor allows capturing of an enormous amount of information about the scene and is possible to parse the same data using different techniques and algorithms to extract different information. The specific implementation of the algorithm depends on the task the camera is set to perform. Restricting the camera to a single perspective can limit the types of information which can be derived, thus the processed data from the cameras are used in conjunction with other techniques, such as moving the camera, providing a modified view through manual distortions or mirrors, and combining multiple sensors.

There are two cameras currently being used for the task of ground texture based visual odometry (Clark & Ferrier, 1992; Hutchinson et al., 1996; Marchand & Chaumette, 2005; Sundaeswaran et al., 1994). The camera's placement is well controlled and is given the task of identifying unique patterns on the ground to be tracked during consecutive frames while the robot moves. The two cameras are synchronized to provide precise 2D pose information including rotation. This implementation, as well as the issues related to localisation is discussed in more detail in chapters 5 to 7.

To observe the environment that is being explored, a single camera is mounted at the front of the robot. This configuration best mimics the visual sensors of most biological systems as the reflected light from the obstacles are observed in the direction of the traversal. The camera is also capable of panning by being mounted on a servo motor, which is currently shared by the sonar module. The rotation allows more precise control over the orientation of the camera over rotating the entire robot. The details of this module is discussed in chapter 10, where it attempts to identify landmarks and meaningful regions of interest in the environment to be integrated to the rest of the knowledge base that is being built up from the sensors (Astigarraga et al., 2004; Mata et al., 2002; Nehmzow et al., 2000).

The last module makes use of a reflective dome to distort the incoming rays to provide an omnidirectional view of the environment (Huang & Trivedi, 1998; Ishiguro, 1998; Nayar, 1997; Yagi, 1999). This module is mounted at the top of the robot and allows the simultaneous view of the immediate surroundings by using a natural compression in the density of the captured image through changes to the viewing angle per photo-sensor ratio. The use of the omnidirectional camera is detailed in chapter 11, where it assists in the scene analysis using its unique characteristics to simultaneously view a larger surrounding area.

## Chapter 4 – Processing

The degree of successful operation for every mobile robot system depends on the appropriate processing and analysis of the sensory data, as well as the internal representation constructed to model the states of the surrounding environment. These form the basis for almost every mobile robotics research, as they attempt to improve the data that are captured from the sensors and represent it in a more precise and meaningful way. To perform these processing tasks, a processing unit is required to efficiently interpret the sensor data and generate the appropriate information required for the given task. It must also consider the relaying of messages between various modules to combine the knowledge to improve and enhance the information.

### 4.1 Processor selection

The key element to carrying out the various tasks is the processing unit that is capable of using the various attributes about the environment and converting the data to a more informative value. Rather than dealing with the intricacies of the processing unit, considerations were made primarily on the ease of use, availability and extensibility in terms of both development and maintenance requirements.

#### 4.1.1 PIC

The simplest type of the processors used on mobile robots is the micro-controller that are used directly on the controller board for manipulating with the sensor at a very low level. The PIC micro-controllers are well suited for highly specialised tasks, as the limited resources and the small set of operations allow for well optimized operations. Although they do not allow for some complex algorithms to be implemented, especially those requiring large memory footprints, the processor has an easy learning curve as the chip is commonly used throughout the course work of undergraduate students.

#### 4.1.2 Laptop computer

While using a specialised processor for a specific task allows for a very efficient processing, it does not provide the flexibility compared to a generalised processing unit, such as that in a personal computer. For an experimental platform, it is advantageous to include the general purpose processing device to simplify the integration of peripherals and the availability of software packages due to the device drivers that are pre-written by the hardware manufacturers. The large range of additional components can allow simple and isolated experiments without the hassle of defining the complex interfaces, while the availability of development kits and libraries allow for accelerated development process when enhancing or extending the system.

### 4.1.2 Laptop computer

Due to the limited space and power available on the mobile platform, it is not feasible to place a desktop computer on the robot. Typical implementations make use of an on-board laptop computer or a message relaying device to off-load some of the non-time critical processing to a nearby computer not on the robot. The laptop computer often acts as the central hub for the various modules to simplify the communication, while providing a greater processing capacity for the robot than the microprocessors.

### 4.1.3 Off-board computer

As mentioned above, an alternative to using a laptop computer on the robot is to relay the data gathered by the sensors to and from a more powerful computer elsewhere. The additional processing power provided by a scalable computer system can greatly increase the robot's capabilities. However, this requires the mobile robot to be equipped with some sort of communication mechanism to transmit the messages as well as requiring an extra consideration to allow for the increased latency in transmitting the messages.

When using an off-board computer to process the sensor information, the benefits of having a central hub on the robot fade and the interfaces between the modules and hardware become a significant issue. This means a dedicated controller board with the appropriate device drivers must be designed and built for each sensor, making this a costly requirement. For this reason, many experimental mobile robots make use of a laptop computer which is also equipped with communication capabilities to relay the process intensive and non-time critical tasks. This hybrid approach combines many of the benefits, but must consider the overheads in splitting the task and also the costs in the actual laptop computer if none are already available.

## 4.2 Communication approaches

When using a tethered approach to link between the robot and the externally located processing computer, the mobile robot becomes quite constrained in its motion as the traversal distance and the possible paths can be severely limited, as well as potentially blocking the view of the sensor signals. Using a wireless mode for communication, it is possible to avoid many of the above issues, but is more prone to loss of messages. Within the robot itself, the use of the cable allows fast and reliable connection between the different modules. Since most of the components on the robot remain fixed or move in a controlled and predictable manner, the cables can be placed so that they do not interfere with the functionality of the other components.

The commercially packaged devices, such as the webcams, come with standard interfaces which can easily connect to other devices. However, many of the other devices require a separate communication device to be attached so the transmission of data can occur to and from the devices without the hassle of changing a lot of the hardware if different modules are connected to it in the future.

When transmitting the data between the different modules, it is important to understand the meaning of the values that are being passed around. This means a certain degree of coupling is required, such as the establishment of protocols and the

## 4.2 Communication approaches

use of consistent units. Since each module is assigned a processing unit, the raw data from the sensors can be converted to an appropriate unit using hard-wired mapping functions. Where there are multiple processors present before the data is passed along, it is possible to leave the conversion until the end to maintain the original data. This late conversion allows certain modules to focus on the conversion and leave the data generating modules to be implemented independently of the whole system. The layer of processes also allows the reuse of algorithms or to view the same data through multiple perspectives which can promote the derivation of new pieces of information. However, the layers can sometimes contribute to excessive coupling and redundancy thus it is important to consider the interfaces between the modules with respect to the given tasks before data is passed on.

### 4.2.1 Processor to sensor communication

The tasks for a given processing unit can vary from a simple sending of pulse instruction to a motor to a more sophisticated feature extraction process. When commanding the various sensors to operate, the processor's task is a matter of converting the high level command, such as “rotate the robot by 45 degrees to the right” to the appropriate command depending on the protocol being used, usually by combining multiple simple commands and operations in a sequence. The process eventually reaches the low level device, where the command is translated to the appropriate instruction for the hardware using the known or calibrated attributes.

The high level commands are typically derived after combining the data from multiple modules and a decision is made based on the task, the state of the environment it has sensed, and the strategies that have been put in place for the current situation. The generated commands are then passed on to the appropriate modules while certain attributes may be filled in by other modules along the way. Although it is easier to visualise the command being carried out through physical modules, a significant number of commands are processed internally in software within the physical devices to reduce the inter-hardware transmissions.

Many of the commands that are generated from a decision which result in mechanical changes are motor based, as most of the sensor modules do not require an explicit high level command to operate. Instead, they are continuously activated and the latest data is relayed whenever another module requires the information. This allows the sensor modules to quickly react to environmental changes instead of having to wait for the chain of processes between modules before events are handled.

### 4.2.2 Sensor to processor

When the sensor generates data, the value represents some device specific reading which must be converted to a higher level concept, such as the time taken or the strength of the received signal. Depending on the task of the module, the information can be converted even further to simplify the process of the subsequent module, possibly resulting in the loss of some information during the conversion process. The information carried to the central processing module, if present, must collate the information from various sources thus requires a set of shareable attributes to be used.

## 4.2.2 Sensor to processor

Without the multiple layers to convert the data to the appropriate format, the central unit is placed with the burden of converting the data to a uniform representation. This requires the central unit to know the type of data being used for the other modules, thus requires constant modification if new modules are introduced. For this reason, it is advantageous to delegate the conversion of data to the other processing units and define a global unit for the other modules to adhere to. Not having access to the raw data is often not a critical issue, since the conversion can often be reversed as long as none of the data is lost.

## 4.2.3 Compression

When transmitting the data across to a different module or system, the data must be formatted according to the transmission protocol, which can sometimes alter the original information. This is often the case with large volume of data producing devices, such as video or audio capturing sensors, which must be compressed before being transmitted to another module. This compression is often necessary due to the limitations in the communication channel and is achieved by discarding or merging portions of the data. The criteria for how to compress the data differs significantly between protocols, thus post-processing algorithms are sometimes required to recover from the degraded information if explicit control of this process is not available.

## 4.3 Current set up

Due to the incremental development of the mobile robot, some of the modules have been layered to simplify the interface between the high level processing unit and the low level hardware interfacing modules. The motor, sonar and the IR sensors are all combined through custom built controller boards which are equipped with variants of the PIC micro-controllers to relay the commands to activate and control the attached devices (Bologiannis et al., 2003; PIC-Servo / PIC-Enc, 2005; PIC-Servo Board, 2005). The sonar and motor modules have been implemented on separate circuit boards and are connected to the main controller board using ribbon cables and a RS485 connection, while the IR module is integrated directly with the main board. The main controller board collates the data from multiple modules and relays messages to the laptop computer through a RS232 connection.

### 4.3.1 Commands to the low level modules

The main controller board interprets the high level commands from the laptop computer to the appropriate units and measurements for the other modules. There are several commands defined in the protocol for various operations to allow the re-use of utility functions and to simplify the view from the laptop's perspective. These commands are listed in table 4.1, which show the currently implemented high level operations. Note that the units for velocity and acceleration depend on the variable characteristics of the robot, such as the weight and battery level, thus is never precisely defined. Some of the commands are not utilised as the high level decision process does not include certain operations or are there for debugging purposes.



### 4.3.1 Commands to the low level modules

Table 4.1: Commands for the controller board.

Command	Argument	Description
n	-	<b>Silent:</b> Toggles sending of messages to the laptop.
i	-	<b>IR:</b> Activate the IR sensors immediately.
c	-	<b>Chirp:</b> Activate the sonar sensors immediately.
y	Angle (degree)	<b>Yaw:</b> Rotation of the sonar servo.
m	Displacement (mm)	<b>Move:</b> Moves the robot forward. Negative for reverse.
v	Rate (unit)	<b>Velocity:</b> Set the maximum speed of the motors.
a	Rate (unit)	<b>Acceleration:</b> Set the acceleration of the motors.
t	Angle (degree)	<b>Turn:</b> Rotates at the center of the robot.
l	Displacement (mm)	<b>Left:</b> Moves the left motor forward.
r	Displacement (mm)	<b>Right:</b> Moves the right motor forward.
e	Rate (unit)	<b>Left velocity:</b> Set the maximum speed of the left motor.
g	Rate (unit)	<b>Right velocity:</b> Set the maximum speed of the right motor
f	Rate (unit)	<b>Left acceleration:</b> Set the acceleration of the left motor.
h	Rate (unit)	<b>Right acceleration:</b> Set the acceleration of the right motor.
o	-	<b>Sonar:</b> Toggles the activation of sonar sensors.

Due to the simple process that is required to execute the command in software, the majority of the time consumption occurs while waiting for the signals to be sent and received, as well as the readying of the hardware for reuse. Because of this, the commands are buffered, where the most recent of the same command type is kept and executed when the resources become available. Note that buffering too many commands can lead to overflow errors or significant delays in processing. Some of the issues can be avoided using a priority queue, but the task of managing the commands is left to the higher level processor on the laptop instead of the PIC chip on the controller board. For operations that have consequence on other sensor behaviour, such as changing the pose, the older commands cannot simply be overridden. This can be dealt with by merging the multiple commands, buffering them infinitely, or by responding to the command generator that the previous command has not been acted yet, thus no further command can be sent.

Time consuming operations, such as the motor motion commands, do not have to wait for the completion of the command before resuming control. Instead, the motor motion is passed onto the controller board of the motor and an interrupt driven flag is checked to note any changes in the state before sending a new command.

### 4.3.2 Data from the low level modules

After the commands are carried out by the individual devices, the acquired data are converted digitally and sent back to the main controller board. This is then collected together and passed back to the laptop computer for higher level processing using the serial connection as a single message. As most modern laptop computers lack a serial connection interface, a serial-to-USB converter has been put in place.

### 4.3.2 Data from the low level modules

The use of the converter has not introduced any limitations to the performance or required the protocol of the commands to be changed.

The data rate of the serial connection between the main controller board and the laptop computer is currently set at 115200 bits per second, which is adequate for the amount of messages being transmitted. The data is currently transmitted in ASCII format for simplicity in the debugging process, but does not contribute to a large overhead in the data size due to the small numbers being represented. The current data format combines the latest sensor data from the sonar sensors, IR sensor array, motors and the timing information from the central board into one and sends it with delimiter characters to allow easy parsing of the message. Figure 4.1 illustrates the protocol with the size of each attributes in bytes when using the binary mode.

Time	Left motor Position	Right motor Position	Sonar servo Position	Left sonar Time of flight	Right sonar Time of flight	IR Distance
4	4	4	1	2	2	12 x 2

Figure 4.1: Messages protocol for binary mode.  
The numbers represent the number of bytes used.

The baud and the content of the message allows for approximately 140 of these messages to be sent in one second. Since this is much greater than the data acquisition rate of the devices, no compression mechanisms are implemented. If the inclusion of additional modules in future implementations result in the need for compression mechanisms, the messages can be converted back to the raw binary format with well utilized data types for each of the values. Another plausible approach would be to split the message and varying the sensor updates depending on the requirement or the individual data acquisition speed of the sensor. This allows each of the modules to operate at their own speed and report the new reading when they become available, rather than reporting an out-dated value due to having to wait for slow operating sensors.

### 4.3.3 Processing on the laptop

The laptop computer acts as a hub for several modules, as it allows simple interfacing with many off-the-shelf devices, as well as the custom built controller boards. Currently, the four webcams and the main controller board are connected to the laptop, all via USB connection. The captured data from the devices are parsed and interpreted individually by the appropriate modules and the extracted information is passed onto the core processing module which combines the information to form some knowledge about the environment, as well as managing the sequence of events in a proper chronological order. Three of the four cameras are also fitted with microphones, but are not used as no audio processing module is implemented. This is mainly due to the limited information an audio data can portray about the environment. The availability of the audio information is beneficial when the robot is under voiced control, as it allows the robot's operator or a speech recognition module to respond to voiced commands from on-lookers, as well as the

### 4.3.3 Processing on the laptop

implementation of speaker localisation techniques when multiple microphones are used (Schauer & Gross, 2001). By combining a speech synthesis module, the robot was successfully deployed as a tour guide for the university's open day by manually controlling the robot from a remote location.

#### 4.3.3.1 Cycle speed

The cycle speed for the main processing module depends greatly on the amount of useful information that can be extracted from the other modules. To allow consistent performance, the cycle speed should be set to the lowest common multiple of the capture rate of all the modules to allow the latest data for any inter-module dependencies. However, this can result in a significant bottleneck as it awaits for synchronisation between the modules. The fluctuations in the sampling intervals from spikes in the processing load and error handling mechanisms can contribute to potential delays, thus causing even further delays in the processing of the data. This approach thus requires extremely high processing speed in each of the modules, or perhaps the inclusion of time-out mechanisms to guarantee a regular interval.

Another strategy to overcome this issue is to set the cycle speed high enough to suit the most time-critical module, then buffering the latest information from the other modules or approximating the current information from trends seen in the past. The predictions can introduce some overheads from the maintenance of past measurements and when determining the actual trends, but can allow for smoother transitions of values from slower modules. Note that this approach can often introduce errors as the trends cannot accurately predict the current state of the environment.

Using the buffered approach, the latest reading from each of the modules can update its buffer entry when ready, possibly with a time-stamp information. Note that the time-stamp information may not be available to the central module, especially if the updates are initiated by the central module. This is because the information may not be extracted or be available. The time-stamp information is maintained by the other module or sent along with the other messages when the central module requests the updated information. This information is vital for detecting duplicates in case the updates did not occur at the expected time, which can cause the prediction based algorithms to misbehave, and to correctly sequence the messages at the proper interval.

Since it is desirable for the main cycle to operate slightly faster than the fastest module, such that the data is not skipped, many of the information being used will be out of time. This allows the cycle to be delayed slightly while it waits for the values from the sensors to be updated. At present, the sensor information from the custom controller board has the fastest data update speed, at approximately 35 messages per second. This does not cause significant issues if some information is lost, as it reports the state of the environment, which is continuous. This is also because the time-stamp used by the central module is also sent along with the message. The messages are parsed and stored in a buffer, which is where the latest information is maintained.

The webcams, on the other hand, which operate at a maximum speed of 30 frames per second, requires the continued sequence of data due to many algorithms requiring transitional information between the captured frames. By making sure that the speed

#### 4.3.3.1 Cycle speed

of the central module is slightly faster, it is able to guarantee that no frame is lost, but duplicate frames must be detected and removed. This process is described in detail in chapter 5. This differs slightly from the sampling theorem, which is interested in the reconstruction of the original continuous signal using infinite number of samples, while this approach is only interested in accessing the continuous stream of frames (Shannon, 1998). By observing the rate of duplicate frames being detected, the cycle speed can be dynamically modified to suit the current processing load.

Although the ideal cycle speed is slightly above 35 frames per second, the addition of extra modules will eventually cause the reduction of the cycle rate to a much lower value. For this reason, certain modules which do not require immediate interactions with another module can perform some of the processing on a separate processor, such as a designated digital signal processor for parsing the image frames. It may also be feasible to off-load some of the non-time critical processing to an external system using wireless communication available from the laptop computer in future implementations.

Since there is only one laptop computer mounted on the mobile robot, there are hardware limitations in terms of the number of devices that can be physically attached. The laptop computer and the devices currently communicate through the USB connection, thus is limited by the number of available USB ports. Although it is possible to increase this by using a USB hub, the net data rate of the USB connection still remains the same due to the sharing of the bus. This causes the data to be delayed based on the priority algorithm defined by the operating system and the bandwidth requirement of the connected devices. Using a faster bus can control this, but care should be taken to arrange the devices such that devices that require high number bandwidth are not using the same shared bus as another high bandwidth devices.

#### 4.3.4 Off-board communication

Although communication to an off-board PC is possible using the built-in wireless card on the laptop computer, the current system does not make use of external processing units to avoid the extra latency and complexity introduced from distributing the processing task and to synchronise between the multiple processing units.

One potential use for the off-board computer is to use it for a data storage unit for the captured information. Since the current processes discard all of the old sensory data, it loses the useful scene information which could become useful for an alternate processing algorithm developed in the future. Maintaining the sensor data also allows manual intervention to view and correct any issues with the automated processes (Graves et al., 1992).

Another use for the communication mechanism is the interactive behaviour mentioned earlier by manually controlling the robot at times. This allows for a much simpler model of interaction than having to chase after the robot to issue special commands and also avoids the driver from being included in the scene analysis process.

## 4.4 Summary

### 4.4 Summary

Figure 4.2 summarises the modules, the hardware, and the connections involved in the current mobile robot platform, where blue represents the hardware and purple represents the different modules. The green lines represent the physical connections between the devices, while the orange lines represent the inter-module connections. As the diagram shows, the design is heavily dominated by the physical link requirements, which places a burden on the laptop computer to perform the majority of the process intensive operations. This will require modification in the near future, such as by developing dedicated localisation hardware or by off-loading portions of the mapping module to the controller board. It is also possible to simply use a more powerful laptop computer in the future, which will allow the majority of the system to remain a generic platform for development of modules.

The core module currently coordinates the operations on the laptop by maintaining an adaptable scheduler to initiate the other modules at the appropriate intervals, with the exception of the controller module. This is achieved by observing the execution times of the scheduled tasks to modify the waiting periods. Due to the heavy load on the processor, the waiting time is almost non-existent and requires some of the process intensive modules to reduce the data rate for it to operate without skipping some data. Currently, this is done by reducing the capture resolution of the cameras, disabling the functionality of the microphones and also the archiving module.

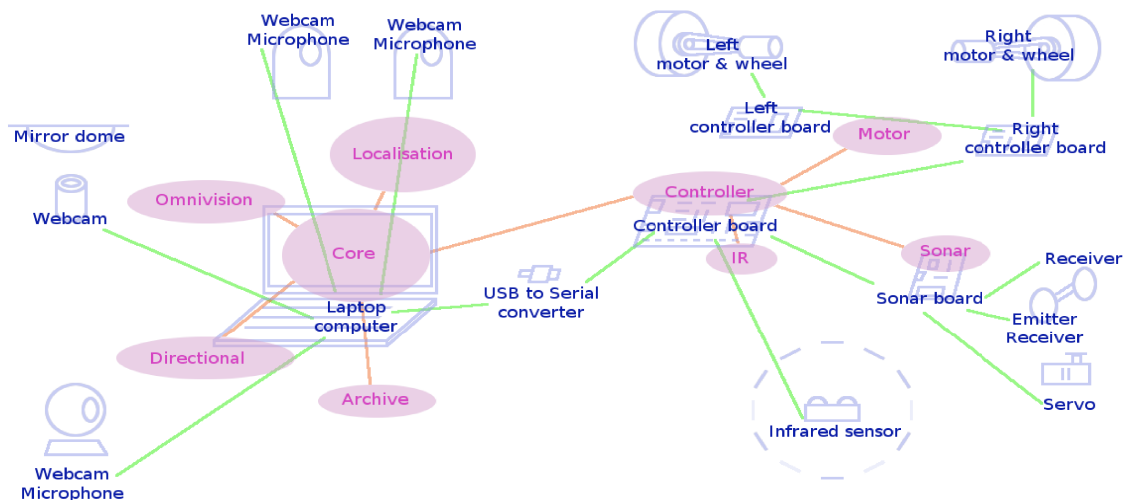


Figure 4.2: The components within the mobile robot system.

The blue objects represent the physical components, the green lines represent physical connection between the physical components, and the purple ovals represent the processing modules, while the orange lines represent the communication between the processing modules.

With the processing issues and the mechanical components involved in the mobile robot analysed, development of data processing algorithms can be carried out. The clear definition of the platform allows the grounding of measurements and algorithms as the two components form a tight coupling to better utilize the capability of each other.

## Section 2 – Localisation

*“Counting steps tell me how far I want to have moved, but the actual question I need to ask is, where am I now?”*

For almost every mobile robot system, it is a vital requirement that the robot know its current location. The locality information can vary in form from absolute coordinate points based on some pre-defined coordinate axes to a high level concept, such as “corridor” or “office room”, which allows it to carry out various navigational and exploration tasks that are appropriate for the immediate surroundings. By knowing where the robot is, it is possible to make informed decisions about where it has and has not been to, as well as where it needs to go. This allows to plan paths to goals and to analyse the environment using the correct perspective (Boudiher et al., 1998).

Localisation for the robots, which is often referred to as *pose maintenance*, can be described as the process of deriving the spatial position and orientation with respect to some representation of the environment. This process is one of the major areas of research in the field of mobile robotics, as it plays a major role in the functionality and reliability of the interactive system. The field is often combined with other research areas which focus on the use of specific sensors and scene analysis techniques to enhance the localisation ability.

The localisation process typically involves the use of open or closed-loop control to accumulate or derive the pose based on the sensor readings and knowledge about the environment. The correlation between the sensor generated data and the internal representation of the environment, whether it be purely theoretical or generated from real sensor readings, allows the pose to be derived when they are combined together (Crowley et al., 1998; Mackenzie & Dudek, 1994; Wolf et al., 2002).

Using the *open loop control* approach, the robot calculates the current pose based on predetermined motion models and the motion commands executed by the motors. This involves the use of forward kinematics and the locomotive characteristics such as the wheel dimensions and gait patterns (Alexander & Maddocks, 1989; Bloch et al., 1996; Chakarov, 2006; Crowley & Reignier, 1992). Although this approach is very simple to implement it relies heavily on the accuracy of the motion model, the consistency of the environment, and the robot's characteristics during the operation, as no feedback information is provided to the robot.

An approach which uses feedback information from sensors, called the *closed-loop control*, requires a much more sophisticated algorithm to correctly interpret the gathered data to derive the current pose of the robot. This process requires an understanding of the various sensor behaviours and the appropriate algorithms to combine the information by correlating with the internal representation of the environment. This allows for the grounding of the sensor readings, such that internal representation can be extended or improved upon (Jensfelt, 2000; Mark et al., 2002).

The correlation process between the sensor readings and the internal representation of the environment can vary significantly between the techniques, but

typically involves the identification of a limited number of regions within the scene deemed as interesting and constrained (Bourque & Dudek, 1998 (b); Bourque et al., 1998; Harris & Jenkin, 2001; Nayar et al., 1994; Sim & Dudek, 1998; Thompson & Pick Jr, 1993), such as the appearance (Ben-Arie & Wang, 1997), to be used as reference points. The use of the significant features within the scene allows for a reduction in the search space when matching the observed scene with the existing representation, as well as simplifying the correlation process when the same feature is encountered (Bennewitz et al., 2006; Davison & Murray, 1998; Se et al., 2002 (a); Thompson et al., 1993; Zhang et al., 1994). The selection criteria for these interesting areas can be derived from pre-determined or adaptive attributes which help distinguish itself amongst the other, repetitive and uninteresting, data (Marsland et al., 2001).

The field of localisation is sometimes categorised into two levels of perspectives and difficulties. *Local localisation* refers to the pose maintenance with respect to the current view of the world to the robot, while *global localisation* is concerned with pose maintenance on a larger scale, typically involving minimising errors in correlations between the states produced by the local localisation (Se et al., 2002 (b); Simmons & Koenig, 1995). Although both areas are ultimately concerned with identifying the pose of the robot, the differences in the scope means different inputs, algorithms, and considerations are required to achieve their respective goals. Typically, the two areas are used in conjunction with each other, in that the global localisation makes use of the results and data gathered from local localisation to assist in disambiguating the pose, as sensors with limited ranges are used.

This ability is often trivial to many biological systems (Redlick et al., 2001), but implementation with very limited number of specialised sensors, the low precision which can be distinguished, the lack of knowledge about the environment, as well as the lack of optimised algorithms integrating the various sensor readings make this process a very challenging task for man-made systems. The overall goal is to construct a system, including both software and hardware, which is able to accurately identify the robot's current location with minimal materiel and processing cost. This often means using existing and affordable sensors to minimise the equipment costs, but developing highly specialised algorithms using a multitude of techniques and algorithms, and also by fusing the data from multiple sensors. There is also scope for simulated environments, which focus on the development of the algorithms, as well as the construction of realistic virtual environments.

Some of the popular implementations of localisation algorithms include the use of purposely designed markers such as signs and bar codes, known scene characteristics, identifying distinguishable features on the fly, or a pose detecting device such as a GPS or a compass (Blaer & Allen, 2005; Bulusu et al., 2001; Guibas et al., 1995; Merke et al., 2004; Shen & Hu, 2004; Werman et al., 1999). These are often combined with dead reckoning algorithms to calculate the current pose, or to estimate the error rate of the dead reckoning approach (Deans & Herbert, 2000; Duckett & Nehmzow, 1999; Ghidary et al., 1999; Kleeman, 1992).

This section introduces a local localisation technique using visual odometry in an unmarked environment using off-the-shelf webcams (Del Bimbo & Santini, 1994). This strategy aims to improve the precision of the local localisation approach, such that it can improve the accuracy of the internal representation of the environment.

Chapter 5 covers the configuration issues in using the camera on the robot

platform. In chapter 6, considerations to the feature detection process is made for the specific configuration and task for the camera, while chapter 7 investigates the various techniques and validity of multi-camera visual odometry techniques.



## Chapter 5 – Camera configurations

In recent years, the availability and popularity of off-the-shelf visual sensors have increased dramatically due to the improvements in the picture quality and the affordability to ordinary home users. This has had positive influences to and from visual data processing techniques as it has allowed more researchers to develop vision based systems. Although the performance aspect of these low cost devices are inferior to the high end visual sensors, this gap is rapidly decreasing as more commercial backing for low-cost cameras drives the technology forward.

Various different fields within robotics and image processing, such as machine vision, active vision, and feature recognition, all relate back to the task of using these photo-sensor arrays to capture the scene structure to be able to identify information from it. The different areas tend to focus on a small portion of the information, which has lead to very sophisticated algorithms and approaches to be developed for specialised tasks and environments (Dudek & Jugessur, 2000). Although some of the basic principles implemented in the algorithms can be reused in other systems, many actual implementations are well optimised to improve their efficiency.

For the task of local localisation, there are several approaches in existence which make use of visual sensors (Beauchemin & Barron, 1995; Corke, 1994; Davison & Kita, 2001). Many of these approaches capture as much of the scene then apply filtering algorithms to extract the desirable aspects within the view (Basri & Weinshall, 1993). The conversion of the 3D environment into a 2D representation introduces ambiguity due to the lack of depth information that can be captured. However, by acknowledging this limitation and utilizing the characteristics that are naturally present or known, the image processing algorithms can be greatly simplified (Lowe, 1987). It can also decrease the processing time and the amount of errors being introduced from the lack of the transformation processes to convert the data into the desired representation (Black & Anandan, 1991). This simplification can be achieved by constraints to the observed scene to a 2D surface, which may be difficult to achieve, but also allows for the increase in the accuracy and reliability of the data as long as the constraint can be maintained (Horn & Schunk, 1981).

A commonly seen use of the above approach is in the optical mouse, which uses a specialised image sensor to measure the displacement of the ground texture from a predetermined perspective (Ng, 2003). An approach which makes use of a similar idea, but using a webcam instead of the purpose built system, is investigated for applicability for mobile robot localisation. Contact based distance measures do exist, but can be influenced by commonly occurring errors like slippage, inconsistent wheel odometry on various surfaces, and the wearing of the system. The vision based approach, on the other hand, allows for a more consistent behaviour on different surface types as long as the camera to surface configuration remain known.

The precision which can be achieved by the optical mouse is incredibly high due to the very short focal distance and the high sampling rate of image capturing process. But the severe constraint on the allowed surface types limits the usage to a

## Chapter 5 – Camera configurations

very small range of environments (Ng & Carne, 2007). The use of a webcam can overcome some of the issues by allowing small variances in height and a larger viewing area at the cost of the loss in precision and increased processing time due to the additional image processing algorithms required to handle the increased volume of data. Figure 5.1 shows a typical set of captured images from a ground pointing camera at different resolutions taken from 10 cm above the ground.

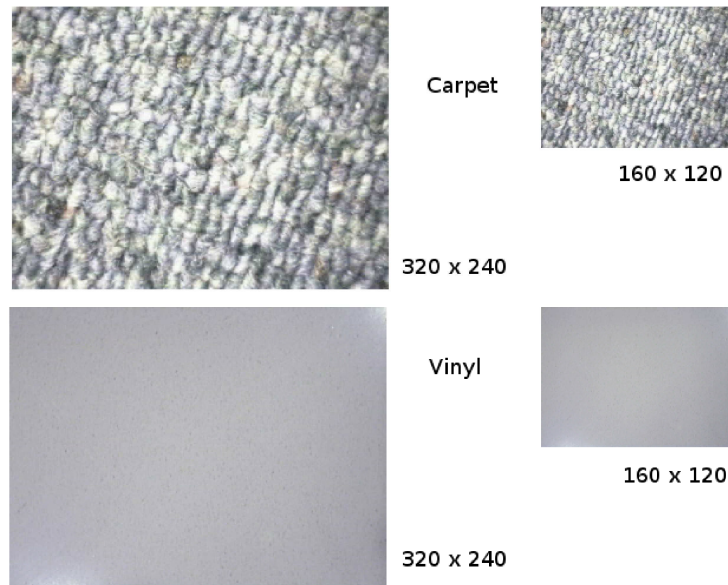


Figure 5.1: Snapshots of ground texture at multiple resolutions. The top two are taken of carpet flooring, while the bottom two are of vinyl flooring. The left column were taken at a resolution of 320 x 240

### 5.1 Camera settings

One of the most important constraints placed by the proposed approach is that the surface being observed must remain flat. To achieve this, the camera must be placed steadily and accurately as this controls the precision which can be achieved by the approach. It also indirectly contributes to other issues such as the illumination of the surface as the camera sensor is a passive device, frame rate for controlling the smoothness of the transitions between consecutive images, and the focal distance, which corresponds to the tolerance in height changes. These issues are considered in more detail from a practical perspective by observing the camera and robot characteristics on various realistic ground textures.

#### 5.1.1 Mount position

In most visual processing algorithms that observe the scene from an arbitrary view point, the transformation processes must convert between the camera's view and the coordinate system used to represent the environment. A calibration process and feedback sensors can be used to detect the changes to the camera's orientation, known as *ego motion*, to derive the necessary transformation matrix (Faugeras et al.,

### 5.1.1 Mount position

1992, Chum et al., 2005). However, a well controlled placement, where the coordinate axes overlaps with each other, during the initial mounting stage can avoid a lengthy transformation process if the camera configuration does not change during execution. This allows for a better utilization of the calibration process, which will avoid a complex self calibration process (Lowe, 2004).

Translating the observed displacement to the actual displacement of the camera requires the calculation of the viewing angle characteristic. This, combined with a known distance to the ground, allows the proportional coefficient to be determined between the two displacements. As the horizontal and vertical viewing angle differs for most cameras, both coefficients must be determined by observing the dimensions of the viewable areas at known distances.

For a typical webcam, the viewing angle is around 40 and 30 degrees for horizontal and vertical directions respectively, which only allows for a small viewing area when the camera is placed close to the ground. Adjusting the height of the camera allows the control of the maximum precision and operational speed of the robot. Since each pixel size remains constant, varying the height allows different amount of area being captured through aliasing, which contributes to the precision that can be achieved. Since the approach relies on tracking the displacement of the previously observed ground texture, the same ground pattern must also exist in the subsequent frame. Hence, the adjustment of the camera height also controls the maximum operational speed of the robot. Figure 5.2 below illustrates the relationship between the various attributes involved, while figure 5.3 shows the relationship in mapping the observed displacement to actual displacement.  $D_A$  and  $D_B$  represent the distance that is viewable in the image, while  $M$  represents the difference in the distance between the camera and the ground to obtain  $D_A$  and  $D_B$ .  $H$  represents the height of the camera,  $W$  represents the width of the projected image,  $\Theta$  represents the viewing angle, while  $I$  represents the position of the point of interest on the projected image.

$$H_A = M * D_A / (D_B - D_A) \quad (5)$$

$$H_B = M * D_B / (D_B - D_A) \quad (6)$$

$$\Theta = 2 * \tan^{-1}((D_B - D_A) / (2 * M)) \quad (7)$$

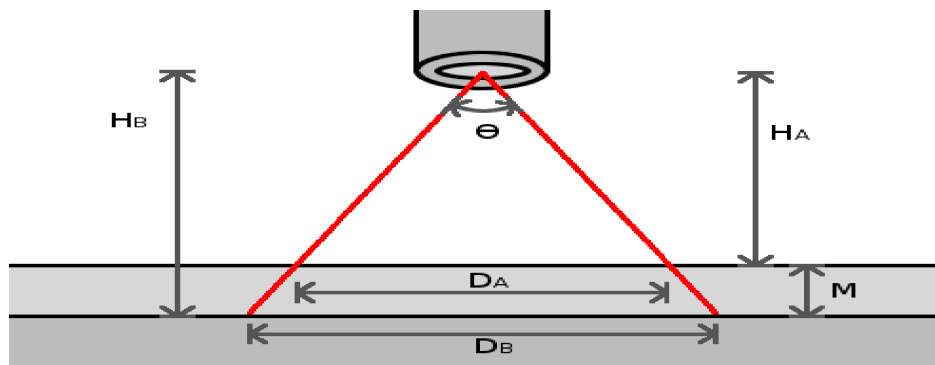


Figure 5.2: Deriving the viewing angle and distance between the camera and the ground.

The attributes of the configuration can be derived by knowing the dimension of the image that is viewable at two different camera positions with known disparity between them.

### 5.1.1 Mount position

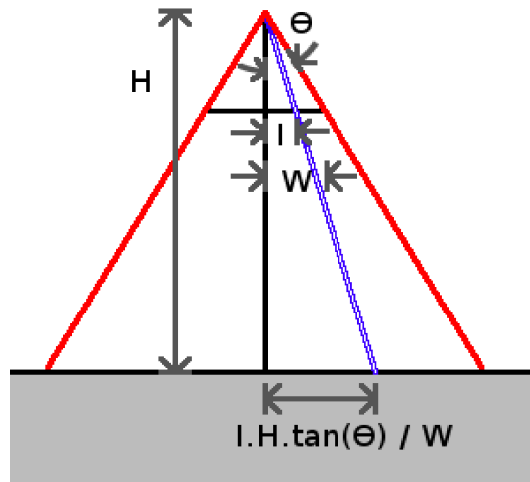


Figure 5.3: Deriving the translation coefficient from pixel coordinates to ground coordinates.

The outer lines represent the limits of the viewing area while the double line represents the location of the point of interest.

A key factor that must also be considered is the processing load of the visual odometry algorithm. Although this issue should be considered for almost every software component, it is also influenced by the physical settings, such as the search area for the texture patterns when the viewing area changes and the necessary modification of the feature size to account for the change in the level of detail when the camera mount height is changed. To optimise the capabilities of the robot, a careful balance between the operational speed, precision requirement, and the processing capacity must be made to utilise the available resources for the given task.

Since the assumption that the camera height with respect to the surface remains consistent can be invalidated from bumps and slopes, the derived displacement can drift from the actual displacement. Simple solutions can be put in place to reduce this effect, such as by using soft tyres to absorb the small bumps, the use of a transparent plate and a spacer to push the protruding objects down or to move it out of the way with a sweeper, and the addition of specialised devices like a gyroscope or focus control on the cameras. However, they do not guarantee the accurate and consistent performance and can introduce other issues like issues with limitation or not being able to traverse over certain damaging surfaces. In the absence of an accurate height change correction mechanism, the original camera height can be used to anticipate the range of the potential drifts. The equations below can be used to approximate the errors for a change in height by a particular amount.

$$G = 2 * I * H * \tan(\Theta / 2) / W \quad (8)$$

$$\Delta G = 2 * I * (H + h) * \tan(\Theta / 2) / W - G \quad (9)$$

$$\Delta G = G + 2 * I * h * \tan(\Theta / 2) / W - G \quad (10)$$

$$\Delta G = h * (2 * I * \tan(\Theta / 2) / W) \quad (11)$$

where  $G$  represents the ground motion,  $I$  represents the motion in the image,  $H$  being the distance from the camera to the ground,  $\Theta$  is the viewing angle,  $W$  is the width of the image, and  $h$  is the change in height.

### 5.1.1 Mount position

Many of the newer camera models are equipped with a software controlled focus adjuster to allow the focusing of the incoming light to the desired location. Since the distance from the camera to the ground remains consistent, this feature does not provide a significant enhancement to the visual odometry technique. It is, however, possible to make use of the focus control feature to make corrections on the camera height, since the image being observed will blur when placed at a non-ideal height. This causes the reduction to the overall strength of the intensity transitions within the image, which can then be used to correct the focal distance. The tolerance of the change in the focal distance is proportional to the distance to the object, as shown in figure 5.4, which is quite short for this type of visual odometry configuration. In the left figure,  $F$  represents the focal distance and  $D$  represents the distance from the lens to the object.

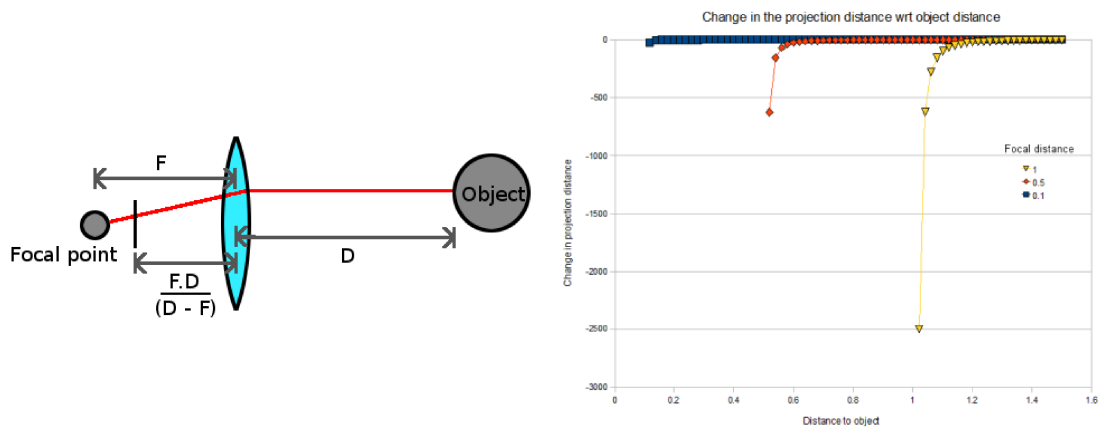


Figure 5.4: Focal distance and tolerance.

The left diagram shows the arrangement of the lens with regards to the object. The right graph illustrates the relationship between the distance to the object and the required change in the projection's position.

Since most webcams are built with the intention of capturing an image of a person seated approximately 1 meter away from the camera, the operational range typically do not allow for the camera to observe something too close. The compact designs of these cameras often limit the lens positions, which control the minimum focal distance and thus the minimum camera height. This threshold is quite significant in some cameras, which can sometimes have a minimum distance of tens of centimeters.

### 5.1.2 Lighting

The passive nature of the cameras means that the device is not able to operate in the absence of a light source. Although the constraint that the height of the camera stays constant indirectly means no obstacles can directly obstruct the view, with the exception of flat obstacles on the surface, this does not stop shadows from extending into the view to modify the appearance of the texture. The main contributor to this is the robot and the camera itself, as the typical ambient light sources are located well above the robot, which causes the camera to cast a shadow directly into the viewing

### 5.1.2 Lighting

area. Figure 5.5 shows a snapshot of a view where a portion of the image is darkened by the shadows from the camera.



Figure 5.5: Snapshot of the viewing area with and without shadows. There is a clear change in the colour where shadows are visible but do not correspond to boundaries that belong to the actual surface.

A simple strategy to overcome this issue is to provide an additional light source with the camera that provides consistent illumination to avoid ambient light changes. Before the light source was used, various characteristics of the light had to be considered, including the direction, the shape of the beam, the brightness, and the colour of the light.

When considering the direction of the light, it was noted that lighting provided from the side allowed for a stronger contrast to be displayed due to the rough surface structure, as shown in figure 5.6. This enhancement of the surface texture characteristics can allow for more uncommon features to be observed, but is limited to when the same surface is observed from the same light source direction. This characteristics means that for a short interval in time which does not involve the rotation of the light source, a light source from the side would provide greater benefit. However, for the feature to remain consistent in between frames over a longer period of time, such as for the purpose of capturing landmarks, the light must provide consistent illumination of the surface from different perspectives. It is also worth noting that the design for the older optical mouse focuses more on the surface roughness rather than the actual colour pattern of the surface. This is changing in recent times, where some of the newer models take into account of both the roughness and colour patterns to allow operation even on flat surfaces.



Figure 5.6: Illumination of the same carpet surface from multiple directions. The left, middle, and right image shows the carpet being illuminated from the left, top, and right respectively.



### 5.1.2 Lighting

Based on the same requirements as above, the surface appearance are required to remain consistent for the particular shaped beam that is used. Using a narrow beam, it is possible to end up with bright streaks or spots, possibly with other interference patterns being displayed from unwanted overlapping of secondary reflections. Sample snapshots of different types of beams can be seen in figure 5.7. To overcome these problems, the light had to be dispersed evenly across the whole view. The initial approach consisted of using a circular array of light emitting diodes (LED) around the camera, but this resulted in spots appearing due to the difficulty in controlling the intensity. An alternate approach of using a bright light source and a scattering surface was implemented next. The set up consists of the bright light source located above the camera shining onto a reflective surface, which was made from a crumpled aluminium foil to scatter the light evenly. The scattering provided by the Lambertian like surface allowed the light source configuration to remain compact rather than the last implementation, which was to move the light source away to disperse the light.

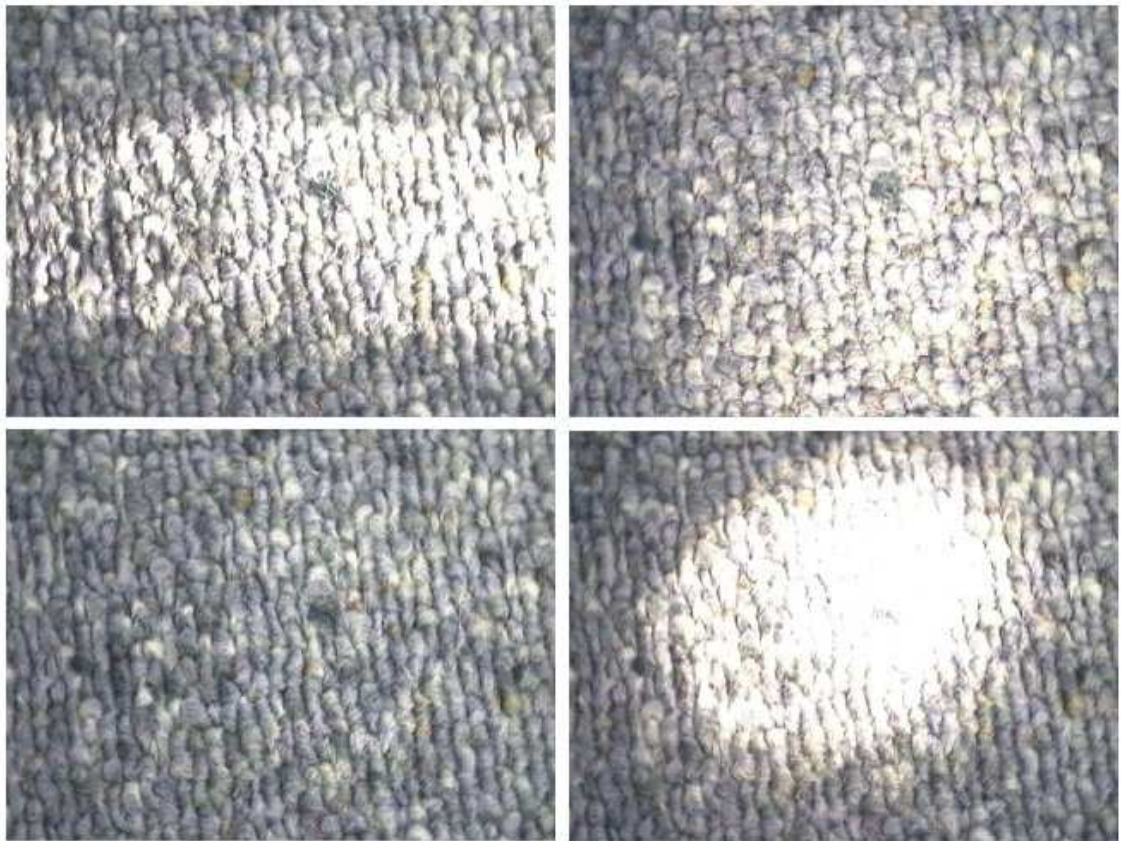


Figure 5.7: Various shaped beams of light.

Top left shows a narrow directed beam from the side, top right shows a wider beam by using secondary reflections from a mirror, bottom left shows a well scattered light to allow uniform distribution of the illumination, while bottom right shows a spot light placed along side the camera.

While investigating the use of LEDs for illumination, various colours were considered, which can be seen in figure 5.8. It is sometimes beneficial to make use of

### 5.1.2 Lighting

a particular light colour when extracting a known coloured component amongst other components. However, without any prior knowledge of the ground texture, or when no typical colour can be dynamically determined, providing a white light source can allow for a consistently performing texture extraction process. As the webcam is able to capture multiple colours simultaneously, the biasing can also be done at a later stage within software. It also allows the inter colour based analyses to be carried out in future implementations, as well as making sure no information is lost due to unexpected surfaces absorbing the particular wavelength. However, using the three colour component increases the processing load, and is typically compressed to a grey scale image in many image processing applications, especially when interested in shapes rather than colours.



Figure 5.8: Snapshot of coloured light sources.

The left, middle, and right image shows the carpet being illuminated by white, red, and green light respectively.

The last consideration which was made with regards to the light source is the brightness of the light. Due to the limited range of intensity readings that can be captured by the camera, as well as the reliance on the exposure time, the brightness of the light source must be controlled to maximise the intensity variance observed in the view. Some light sources can allow the brightness to be controlled, but others require physical masks to dampen the intensity if it is too bright. This can be achieved by placing a semi-transparent material over the light source, which can also assist in scattering the light. As previously mentioned, it is also possible to move the light source away from the surface, but this is often limited by the physical constraints on where the light source can be attached. Instead of modifying the intensity of the light source, the exposure time setting of the camera was investigated in more detail.

### 5.1.3 Exposure time

By controlling the exposure time of the camera, it allows different amount of light to be captured to modify the apparent intensity. Most cameras and their drivers are bundled with automatic exposure control to adjust the shutter speed and gain to suit the ambient light present in the environment. This filter is often applied by using the average intensity of the whole view to shift and stretch the intensity to allow for varying light conditions. However, this shifting causes inconsistencies in the apparent intensity of objects under varying lighting conditions, thus must be reversed when attempting to use the intensity as an identity measure. The exposure time also has the effect of causing motion blur when it is set high, which can be problematic when analysing views of moving objects and when the exposure is set high due to



### 5.1.3 Exposure time

low level of ambient light.

By setting the exposure time to be very short, it can reduce the effect of motion blur, but limits the amount of light entering the camera to limit the richness of the captured texture. This effect can be seen in figure 5.9. With the introduction of the permanent light source, the exposure time can be controlled depending on the flexibility in the brightness level. By setting the brightness of the light to be very high, it also creates a larger variance in the observed intensity, thus must make sure the light is dispersed evenly as possible. By manually controlling the exposure time, it can be reduced to the point of being able to observe the most amount of variance in the ground texture, which can also reduce the artefacts from motion blur while still allowing enough contrast in the ground texture.



Figure 5.9: Various exposure time at constant brightness.  
The exposure time is decreased in the order of top left, top right, bottom left, and then bottom right.

With the exposure time being fixed at a constant value, the camera is unable to make adjustments when the ambient light conditions change, such as when shadows form or the room light being toggled. There is also a subtle flickering that occurs in indoor operations due to the alternating current induced timing differences between the room light and the sampling time of the camera. Although the timing offset only causes a very gradual change in the intensity, the overall effect of this flickering can be quite significant, as shown in figure 5.10. Software algorithms can be put in place to detect these conditions, but a simpler solution is to modify the frame rate of the camera or to physically shield the viewing area, such that the additional light source becomes the sole provider of the light. Completely shielding the light is difficult to achieve since the shield would scrape against the ground, thus a small gap must be made off the floor. As a side note, this shield also doubles as a sweeper to remove light obstacles which may enter within the view of the camera.

### 5.1.3 Exposure time



Figure 5.10: Flickering caused by ambient light. Slight changes in the appearance can be noticed between consecutive frames from the room light.

Other than having discrete intensity readings, the cameras differ from organic visual sensors by not having a logarithmic response characteristic to the light intensity. This results in a very narrow range of intensities that can be observed at once. By applying techniques similar to high dynamic ranging sensors, it is sometimes possible to compose an image which maximises the amount of information by stretching or compressing the inter-pixel intensity differences. When capturing the image with a fixed exposure time, different amount of light allows certain portions of the scene to be captured better than the other. By adjusting the exposure setting for the same scene, it is possible to superimpose and merge multiple images into one, as shown in figure 5.11. This allows better utilisation of the given range of intensities to produce a very information rich image.



Figure 5.11: Merging of the four images from figure 5.9. Superimposing and blending of multiple frames allows the interesting portions of the imaged to be combined and shown in one frame.

### 5.1.3 Exposure time

Tone mapping techniques such as using the median value and using the exposure time when a certain intensity level is reached can be applied, but the shifting of the intensity results in the loss of colour based information and also requires multiple frames to be captured using different exposure settings. Implementing this on a webcam is a challenging task, since the exposure control is not readability available on some cameras; it greatly reduces the frame rate, and also requires the elements of the scene to stay stationary while multiple images are captured.

Implementing this for a mobile robot can potentially be useful if used in an environment where large fluctuations in the light intensities occur and where the focus is on the shape of the features rather than the colour, such as sign recognition tasks in an outdoor environment or rooms with windows on a sunny day for an indoor application. For the task of ground texture viewing, the lighting conditions are well controlled, thus this approach has little applicability.

### 5.1.4 Capture rate

The last camera settings to be considered are the capture rate of the images and the synchronization issues which arise from the use of multiple clocks. One of the key contributors in achieving the high level of precision on the optical mouse is in its extremely fast frame rate, which is typically over several thousand frames per second. The high rate of data allows for very small motions to be observed, which accumulate to a very smooth motion being observed. The increase in the processing load is offset by the small size of the texture being captured, at sometimes around 8 by 8 pixels, and the smaller search area due to the reduced distance the mouse can be moved within the shorter time span.

Indirectly, the capturing rate contributes to the maximum operational speed, as this determines whether a pattern stays within the field of view in the subsequent frame. This relationship is illustrated in figure 5.12, where  $L$  is the width of the area being tracked and  $W$  is the width of the captured image. Ideally, the capturing rate should be set to the maximum possible setting while taking into account of the data transfer rate between the modules and devices. The smoothness provided by the high sample rate allows the transformational changes to the images to remain small and also increases the validity of prediction algorithms which may be implemented (Faugeras & Robert, 1993).

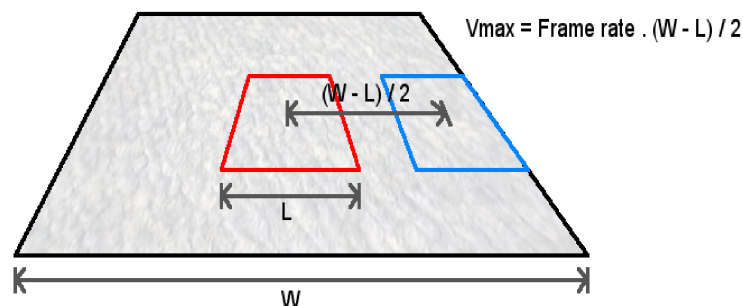


Figure 5.12: Effect of sample rates against the operational speed. The motion of the robot is limited by the sampling rate of the floor texture.

#### 5.1.4 Capture rate

A potential issue in using multiple components is the timing difference between the data capture time and the access time, which can cause the misalignment between the scene and the internal representation or disrupting the behaviour of the devices with manually induced pauses. With the webcams, the capturing process and the reading process are executed independently. If the reading process is slower than the capturing process, the extra frames that are captured typically override the old data to only maintain the latest information. If, on the other hand, the reading process is faster, the old data will be read in. Ignoring this behaviour can cause loss of critical information about the scene such as jumps in motion of objects and incorrect predictions about the object's behaviour.

These issues can sometimes be addressed with the use of precise timers or semaphores, but can suffer from clock drifts over long periods and unnecessary pauses within the process. An alternative approach is to observe that the loss of frames cause unrecoverable errors as the texture tracking would not be able to see the features that have exited the field of view, but a duplicate frame can be easily detected to avoid the errors being introduced to prediction algorithms. With this in mind, as long as the cycle speed of the reading process is faster than the capturing rate, the tracking can still operate with the same level of accuracy at the rate of the capturing process. To detect the occurrence of the duplicate frame, the change in the intensity within the frame can be observed to determine if the image has not been updated yet. This technique is plausible due to the presence of random noise, which causes the image to change even if the camera observes the same portion of the ground.

It is also possible to calculate the clock offsets and the amount of drift, but this depends greatly on being able to know the exact changes in the processing load to anticipate spikes which can modify the timing. Therefore, it is more plausible and reliable to specify the maximum processing load, then deriving the minimum cycle speed of the reading process for the current sampling rate of the camera.

## 5.2 Image and sensor noise

For many years, the field of image processing and robotics have progressed in parallel and under a merged name called machine vision. The discipline attempts to combine the data rich visual sensors, with its plentiful information extraction and processing algorithms, to an autonomous and physical agent for carrying out real vision related tasks. As developments are made in both areas, the integration allows for superior sensor systems to be developed. Rather than making use of high end hardware to perform the image capturing process, there has been an increase in the use of a more economical webcam in its place. The quality of these cameras is constantly improving, but due to the manufacturing process and the components being used, they produce lower quality images.

The quality of the image is limited by the characteristics of the complimentary metal oxide silicon chip, which records noisy data due to the photo-sensor and transistor arrangements. Many of the older model webcams make use of plastic lenses, which purposely blur the image for natural interpolation of the textures. This is done to mask the granularity caused by the low resolution and the poor quality of the sensors. These contribute to the artefacts to the image, but by identifying these

## 5.2 Image and sensor noise

noise characteristics and developing the appropriate filter to remove the effects, the webcams will be able to provide useful and reliable information about the scene.

### 5.2.1 Lens distortion

Due to the warping introduced by the camera lens, the image cast onto the photo-sensor array tends to be distorted in a radial pattern. This is caused by the projection of the Petzval surface to a flat plane. For the webcam, this causes the compression of the outer regions of the image. The effect is sometimes corrected at the driver level software for some of newer models, but in the absence of the de-warping algorithm, the distortion characteristic can be determined by observing a known shape, such as a grid pattern, throughout the whole view and stretching the image until the calibration object can be observed without distortion. This process typically involves the use of algorithms like the Hough transforms, as shown in figure 5.13, to characterise the features observed within the view (Kalviainen & Hirvonen, 1995; Shaked et al., 1994).

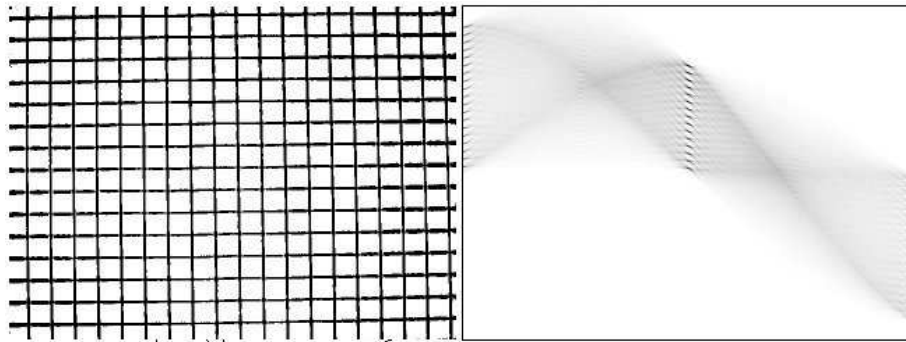


Figure 5.13: Hough transform for calibration.

The straightness of each line is determined through the sharpness of the meeting points of the faint curves.

The cameras being used in the current system showed very little warping due to the manufacturer's lens distortion correction implementation, but the side effect of the stretching caused visible signs of blurring at the outer edges of the image. Attempting to reduce the blurring by applying sharpening algorithms can result in the enhancement of noise generated features or a costly process involving the identification and suppression of the de-warping induced blurring by weighted adjustments to the intensities.

A useful characteristic to know about the warping is the gradual increase in the distortion and blurring from the center of the image to the outer edges. This means that although the distortion is non-linear, the difference in the artefacts between adjacent pixels and consecutive frames is quite small. However, since the effect of the blurring is not noticeable within the inner portion of the image, the outer portion can simply be cropped out from further analyses. The reduction in the overall image size does not cause a significant issue to the visual odometry algorithm other than reducing the range of acceptable motions. This is because of the processing cost involved in searching over a large area, thus only a small portion of the captured image is used.

## 5.2.2 Sensor noise

### 5.2.2 Sensor noise

Although there are always issues with sensor noise with any type of cameras, webcams are notorious for the low quality images they capture. Although the quality depends on many other factors such as the colour richness, refresh rate and lens quality, the amount of noise introduced by the photo-sensor contributes to a significant portion of the problem facing image processing tasks. This noise is generated by the hardware itself and thus cannot be prevented. Instead, filters must be introduced to suppress the noise level.

When viewing the image as a whole, these noises are not immediately apparent as our eyes tend to focus on the larger, semantic information portrayed within the image. However, when attempting to identify and track a particular pattern by considering the intensity reading at each pixel, changes in the appearance can cause problems for the image processing algorithms. The problem is further enhanced by the highly repetitive texture patterns and the lack of variety in the intensity within a single viewing area.

To identify the contributions from the noise, the camera was exposed to a variety of different conditions to isolate and characterise the behaviour shown through the intensity readings. The first measure to be identified was the per pixel based noise level, which is the noise from the photo-sensor sensitivity, interference from neighbouring pixels, and any defects, such as scratches on the lens. The measures that were identified include the minimum, maximum, mean, and the standard deviation of the intensity for each of the pixels as the camera was exposed to several different colours. To minimise the neighbouring photo-sensor interference from occurring, the intensity was chosen to be uniform for all three colour component, as well as throughout the whole image.

Providing uniform intensity was quite easy to achieve for the two extremities, since they just required the elimination of the light source for black colour and saturation by a bright light source for white. When exposing a grey image to the camera, the variation in the detected intensities from other artefacts was distinctly visible, thus the data gathered could not be used to generalise the hardware generated noise for the grey colour.

To gather the typical intensities, the same view was sampled until the mean values converged to a point where the maximum variation could not be distinguished on a pre-determined range. This point can be formalised with the following relationship:

$$\text{Precision} > |\Delta I_{\text{ave}}| \quad (12)$$

$$\text{Precision} > |(I_{n,\text{ave}} * n + I_{n+1}) / (n + 1) - I_{n,\text{ave}}| \quad (13)$$

$$\text{Precision} > |I_{n+1} - I_{n,\text{ave}}| / (n + 1) \quad (14)$$

Where  $I_{\text{ave}}$  is the average intensity and  $n$  is the number of samples. Note that the change in the mean is maximised when the difference between the average and the next intensity is greatest. This occurs when the average is 0 and the next intensity is 1, and vice versa, thus leading to:

$$\text{Precision} > 1 / (n + 1) \quad (15)$$

Since the range of the intensity for most modern coloured devices is  $2^8$ , the



### 5.2.2 Sensor noise

precision, which is defined as the resolution in the pixel intensity, is  $2^{-8}$ . This means the sampling must occur at least 257 times.

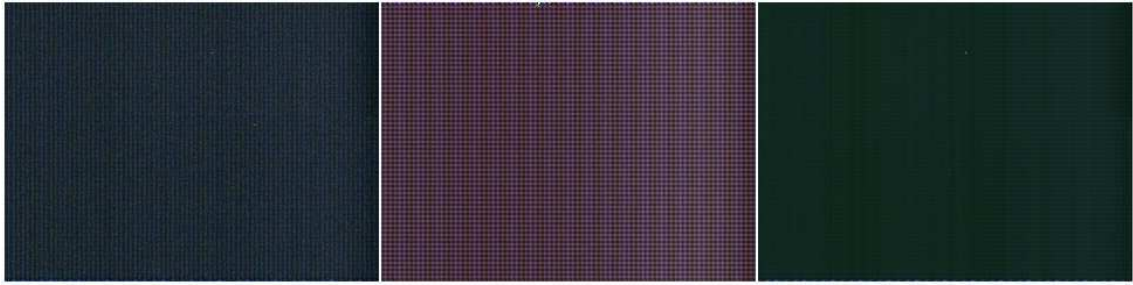


Figure 5.14: Noise level when exposed to a pitch black scene. Left image shows the maximum error reading, middle image is the standard deviation that has been stretched, while the right image shows the average intensity reading.

Figure 5.14 illustrates the noise characteristics detected from the above experiment, while table 5.1 shows the noise characteristics of the image, both for the black scene. The left image in figure 5.14 shows the maximum reading, middle shows the standard deviation, while the right shows the average readings, where all the values have been stretched to enhance visibility. An interesting behaviour which was observed was with regards to the saturation point, as well as the regular pattern in the noise prone areas. When the camera was exposed to the bright light source, every pixel was saturated to the point where no fluctuations could be observed. From a noise removal perspective, this is simply an exception case to be wary of, but from an image processing perspective, a saturated pixel is a tricky case where there is too much light present and the intensity characteristic is lost, thus should be avoided by carefully setting the exposure rate and noting the brightness of the light sources.

Table 5.1: Minimum, maximum, mean, and standard deviation of noise at sample points when observing a black scene.

Colour	Minimum	Maximum	Mean	Standard Deviation
Red	0	55	15.52	8.23
Green	0	41	14.14	6.81
Blue	0	39	9.89	15.78

The noise characteristics observed when no light source was present showed some patterns, which is primarily caused by the photo-sensor arrangement and artefacts from the compression in the codec. The interesting observation which was made was that the minimum values that could be observed for most of the pixels were slightly above zero. This meant that the pixels can not distinguish the intended intensity range. The pattern also showed the different levels of fluctuation. This information allows the formation of a location dependant filter to account for the amount of noise that is to be expected.

Since there was no light source passing through the lens, the obvious regions of irregularity can be attributed to faulty photo-sensors. The amount of noise observed in these areas were significantly higher than the other regions, thus the use of the

### 5.2.2 Sensor noise

pixels should be avoided when possible, such as by cropping or interpolating the neighbouring pixels.

It is also worth noting that some device drivers require sequential distinguishable frames to allow for some of the image adjustment and restoration processes to take place. This meant exposing the camera to a pitch dark scene would halt the camera until a brighter scene is observed. To test for the characteristics of the camera under no light, the calibration process has to be carried out multiple times to allow a portion of the view to be exposed to the light while the remaining region is observing black.

When performing the above experiment for the various grey levels, it was noted that if the obvious scaling effect was ignored around the image, which will be discussed later, and the intensity levels were treated as the mean of what was captured, a significant trend was observed with the amount of fluctuation seen for the various intensity levels. This prompted a more formal experiment involving measuring the noise level for the full grey-scale range. The intensity levels between the three colour sensors were kept as close to each other as possible, but the slight differences may have contributed to some of the inter-sensor interference based noise.

The experiment made use of a grey-scale gradient to note the same attributes as the earlier. Figure 5.15 shows the noise trend for the three different colours at different intensities. The difference between the intended and captured intensity due to the location within the view did not matter, since the intention was to identify the general characteristics of the fluctuation in the intensities.

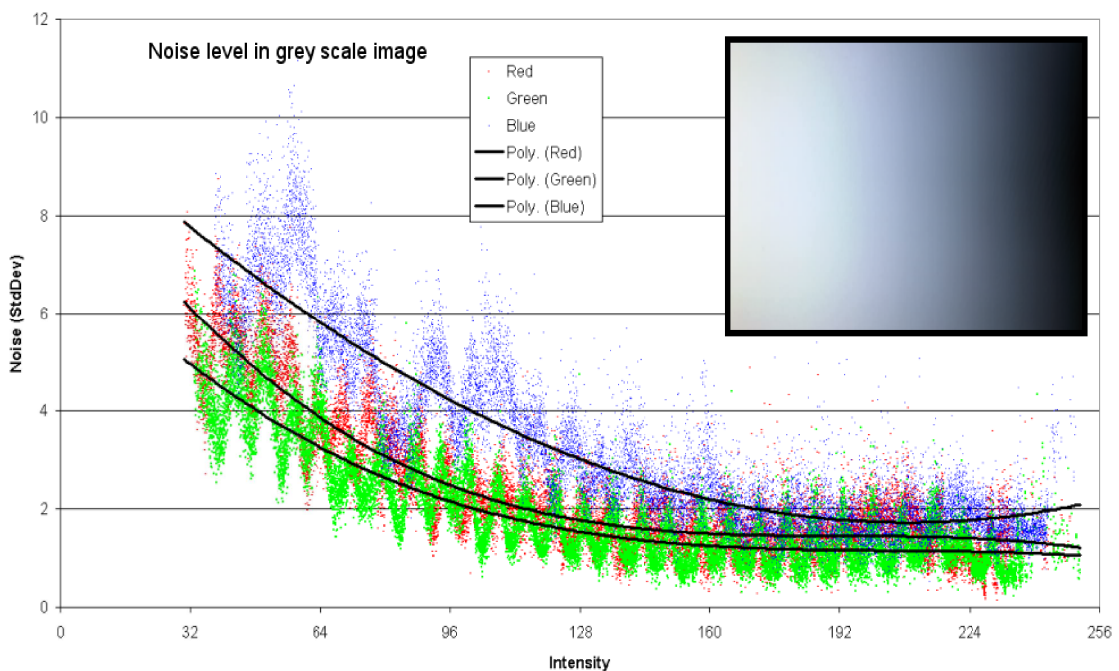


Figure 5.15: Trend in noise for a given intensity. The graph on the left illustrates the reduction in noise when observing a brighter colour. The top right image was used as the calibration image.



### 5.2.2 Sensor noise

Although not all of the intensity was captured, the trend that the noise level reduces with higher intensity value could be observed for all three colour components. The trend showed that lighting up the scene or adjusting the camera setting to capture a brighter scene would result in a less noisy image. However, careful considerations must be made to not saturate the image, as mentioned earlier, as the region will no longer be distinguishable. It is worth noting that the saturation point and the range is dependant on the brightness and gain settings on the camera, thus these settings should be set appropriately for the exposure time being used to maximise the intensity variation that can be observed.

Another distinct feature that is apparent in the trend is the regular wave pattern in each of the three colours. Although this may appear to be a significant artefact interfering with the experiment, it is caused by the hardware related noise pattern encountered earlier for the black image. The effect is distinctly visible due to the alignment of the image being captured, where the grey level is increased in the direction.

During these testing processes, it was noted that the noise does not extend to the whole range of intensities, but fluctuates close to the intended intensity value. This behaviour allows for various noise reduction filters to limit the range of alterations it makes to the captured intensity, as well as providing more capability to derive the actual intensity.

The consideration of the photo-sensor arrangements (Adams, 1997; Hubel et al., 2004) are not discussed here, but can potentially allow more meaningful noise characteristics to be identified, such as explaining the regular noise pattern observed earlier.

### 5.2.3 Colour based noise

When analysing the noise level of a pixel, the three colour components were treated independently for the same source of light. The goal was to allow the characteristics of each colour sensor to respond to the same amount of light being captured by controlling the light source. Ideally, the photo-sensors should not cross-talk with each other, but the sensors will typically be exposed to a wide range of intensities in a natural environment. This can lead to varying interactions between the photo-sensors which require analyses.

An initial experiment involved a very rough measure of the noise characteristics for various colours. This was done by showing an image of the full visible spectrum to the camera and identifying the individual noise characteristics at each pixel. Since each of the pixels was exposed to multiple colours within proximity it was not able to give a clean characteristics of the noise data for a single colour, but an obvious trend was observed, as shown in figure 5.16 below, between the exposed colour and the amount of noise for the colour.

### 5.2.3 Colour based noise

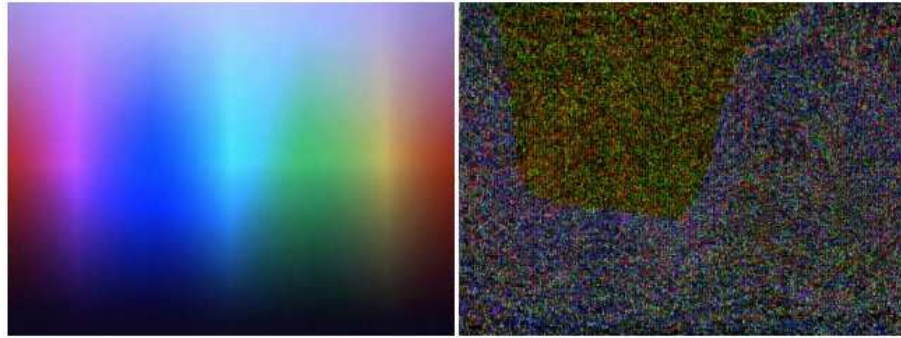


Figure 5.16: Calibration image and the corresponding standard deviation in colour. Left image shows what the camera observed, while the right image shows the relative noise levels.

The trend observed above prompted a more detailed analysis in identifying the noise characteristics for a given colour. Due to the huge number of variation in the possible colours, it was unreasonable to carry out the detailed experiment for each of the  $2^{24}$  colours. To reduce the number of data being captured, while providing reasonable amount of redundancy to account for other noises interfering with the data, the sampling area was reduced to a small square of 14 by 14 pixels, such that multiple intensities could be tested at the same time. When positioning the sampling areas for the camera, the noise prone areas that were detected earlier were avoided. The locations were specified manually as the variation in the other portions of the image did not appear to be significantly different to each other, but a better location could have been selected using simple minimization algorithms. To account for the interpolation from neighbouring pixels from affecting the sampled square, the 2 border pixels surrounding the square were also kept as the same colour. The arrangement of these sampling areas is shown in figure 5.17.

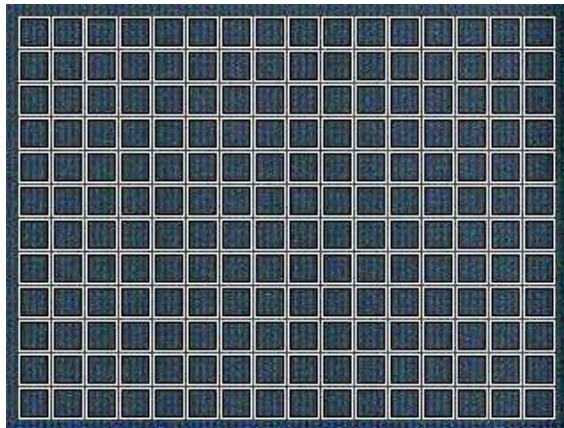


Figure 5.17: Arrangement of sampling squares. The sampling squares are superimposed over the maximum noise image to illustrate reasonably consistent sensor characteristics.

Even with the simultaneous testing for multiple colours at the same time, the number of colour variations is still too large to be sampled individually. Since the trend observed in the simple experiment showed a regular pattern with varying colour, the precision used for the colour was also reduced to a quarter for each of the three colours, which can be interpolated later to fill in the missing values. The trend,

### 5.2.3 Colour based noise

when observed individually for the three colours, turns out to be quite similar to the one found for the grey-scale experiment, which can be seen in figure 5.18, and did not show deviation when varying the other colour components. The trend which was observed in the simple experiment was most likely caused by the by product of observing a highly colour variant scene under irregular ambient lighting conditions and the different thresholds in the sensitivity of the sensors.

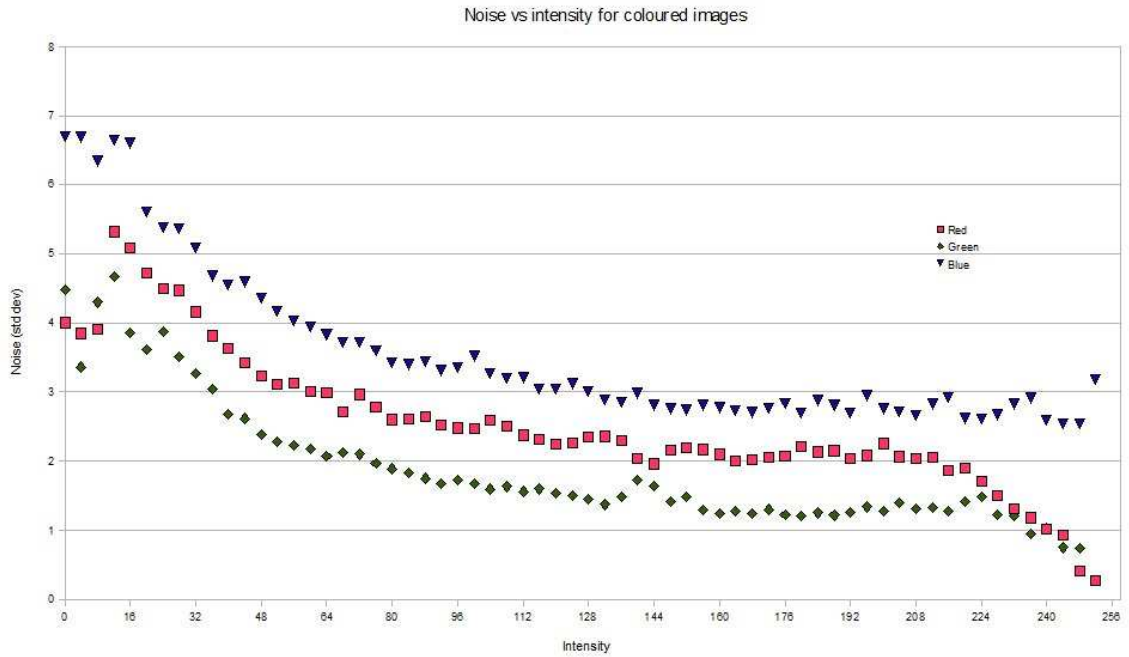


Figure 5.18: Standard deviation in intensity for various colours. The trend observed was similar to that of figure 5.15.

Since the refractive index of the lenses usually depends on the wavelength of the light, the three waves that are detected do not have the same focal length. Whether this is accounted for by the photo-sensors is unknown, but could contribute to the variation in the noise levels for the same intensity due to the blurring and dispersion of the light, especially near the outer edges where the difference in the focal length is greater.

### 5.2.4 Radial intensity shift

When observing the uniform grey image, the captured image showed a gradual darkening as it neared the outer edges of the viewing area. This effect can be attributed to the lens characteristics, interference from reflected rays, and the increased distance and angle from the incidence angle (Basri & Jacobs, 2000). This radial intensity shifting behaviour is heavily dependant on the ambient light that is present and the reflective properties of the observed surface, thus if the light's characteristics, such as the incidence angle and the specular properties of the surface are known, the center point of the this radial effect and the amount of necessary shifting can be determined to smooth out the image.

Since these attributes cannot be known in most scenarios, the plausibility of a generic filter must be carefully analysed in the case that the adjustment is incorrectly

#### 5.2.4 Radial intensity shift

aligned and weighted. Figure 5.19 shows a typical view of a uniformly coloured surface with the radial intensity differences. The left shows the actual image, middle shows the relative intensities, and the right shows the standard deviation. Since the difference experienced between the pixel intensities is most significant between the center and the outer edge of the image, the cropping process, as used earlier, can sometimes be considered as an alternative approach to reduce this effect. The rings indicate the subtleness of the effect, as well as the aliasing of colours that has occurred from the image compression process and the sensitivity of the photo sensors.



Figure 5.19: Radial shift characteristics on a uniform looking surface. Left image is the actual image captured by the camera, middle image shows the stretched image to highlight the difference in the intensity. The right image shows the standard deviation, which clearly shows the formation of bands to group the similar intensities.

### 5.3 Image processing filters

The characteristics that were found during the calibration phase can be used to design the filters to correct any noise in the stream of images. The effects of the noise and the conditions under which the artefacts were introduced were considered to derive several filters.

The filters work on the principle of observing the captured intensity for the pixel of interest, sometimes along with the surrounding pixels, and applying a transformation based on the position, the intensity, and on the change in the intensity between consecutive frames. By adjusting the parameters for these attributes using the characteristics determined earlier, the undesired artefacts can be reduced while minimising the additional artefacts being introduced by the filter, which commonly occur with standard image filters.

The evaluation and validation of these filters are very difficult to achieve due to multiple factors that contribute to the noise. It is also difficult to justify the restoration result amongst many other variables, thus the assessment process is done individually for the artefacts the filter attempts to reduce. Other than the noise removal and the restoration of the intended intensity, the other common attribute considered as part of the evaluation included the amount of artefacts that are introduced from incorrect use of the filter, as well as the speed and memory usage taken to process the image.

### 5.3.1 Range modification

### 5.3.1 Range modification

The first filter to be investigated makes use of the intensity range which was detected for the three colour values at each pixel. Ideally, this range would be a constant value covering the maximum range possible for the camera sensor. However, the observations indicated the potential presence of sensitivity thresholds, offsets, or precision inconsistencies which may be causing the reduction in the range. Since the maximum intensity resulted in a constant intensity reading from the saturation, this adjustment only concerns the characteristics of the minimum intensity value that could be detected.

In the case of a sensitivity threshold, where values that are too low are shifted up from the minimum value, the average values and the standard deviation should support this and be lower than usual as more intended intensities are mapped to the same lowest observed intensity. However, the small minimum value and the much larger fluctuation experienced at the lower intensities meant that the cause of the noise was most likely not due to the above hypothesis.

Assuming that the shifting of the minimum value is caused by an offset, the available range may have been stretched to allow the full span of the intensity range. This requires the use of a lookup table for the offset value,  $I_{min}$ , at each pixel and a scaling function to derive the new intensity,  $I$ , from the captured intensity,  $I_{raw}$ :

$$I = (I_{raw} - I_{min}) / (1 - I_{min}) \quad (16)$$

The mapped intensity is intended to balance the three colour components within the image by stretching the value over the maximum possible range. Disregarding the obvious effects from the radial shifting, the standard deviation for a particular colour should decrease as the colour scales match up. However, the tests indicated that the scaling of the ranges caused slightly larger fluctuations in the intensities, which are shown in table 5.2 for a red dominated, a green dominated, and a blue dominated colour.

Table 5.2: Stretching the range using the minimum intensity value detected.

Colour		Original		Scaled	
		Average	Std. dev.	Average	Std. dev.
Red	R	8.3	4.0	7.42	4.14
	G	73.26	7.58	69.94	7.73
	B	253.29	1.26	253.28	1.26
Green	R	64.0	8.57	63.31	8.68
	G	181.06	13.01	179.7	13.45
	B	111.97	13.55	111.33	13.65
Blue	R	193.98	14.21	193.76	14.31
	G	44.17	5.71	40.32	5.87
	B	18.55	3.55	17.49	3.6

### 5.3.1 Range modification

An alternative use for the minimum value was considered which extends the offset idea, where the minimum value may be caused by the shifting of the representation while the value that is read is correctly scaled. To test this approach, the minimum value was added to the value that was read, encouraging a brighter image. A side effect of this approach is in the shifting of the range from  $[I_{\min}, 1)$  to  $[2 * I_{\min}, 1 + I_{\min})$ , which results in some of the brighter intensities being truncated. The experiment itself was carried out in the same way as the range stretching test, but like the other, resulted in causing a larger fluctuation of the noise.

The findings here indicate that minimum value is most likely caused by internal noise within the photo-sensor circuitry and overlaps with the incoming light, which responds to the higher value. This could also be a side effect of avoiding a pitch black image from being captured which affects some of the automated camera setting control algorithms.

### 5.3.2 Filter selection

Many of the image processing filters in existence defines a generic template for the algorithm and are typically used across the entire image without acknowledging the side effects. By blindly applying these filters, it has the effect of enhancing or suppressing certain portions of the image while introducing artefacts where the filter effect is not applicable. Post-processing of the filtered image is sometimes carried out to identify and remove the artefacts. This is often combined with the analysis stage by setting a threshold criterion to correct the wrongly modified regions. This filter application approach can sometimes be problematic due to the dependency on the initial parameters used to transform the image and also on the threshold value used to distinguish the difference between the true and false positives.

Instead of using a generic image transformation filter, a customised filter based on the pixel and colour characteristics can be applied to avoid the post-processing phase. This will also allow the appropriate amount of weights to be used to restore the image according to the current state of the pixels.

#### 5.3.2.1 Spatial filter

One commonly used filter is a neighbour or spatial filter, where the value of the neighbouring pixel influences the current pixel of interest. In many cases, there is a constant weight factor used when combining the intensity information, but this can be modified using the characteristics found earlier to control the weighting.

Based on a blurring filter, a noise reduction algorithm can be implemented by interpolating the neighbouring pixels to generate smoother transitioned pixel intensity (Simoncelli, 1996). The discrete pixel interval means the transitional trend must be approximated using the surrounding pixels. This can sometimes involve calculating the equivalent to the derivatives by finding the difference between the surrounding intensities and interpolating between them to derive the new intensity. The approximated intensity can be compared to the measured intensity to determine how correct the approximation is and to see if the difference was caused by the sensor generated noise. Since the sensor generated noise causes fluctuations around the actual intensity, the tolerance range, which can be set to the standard deviation or



### 5.3.2.1 Spatial filter

the maximum and minimum range in the intensity measured earlier, can be used to distinguish actual changes in the view or noise.

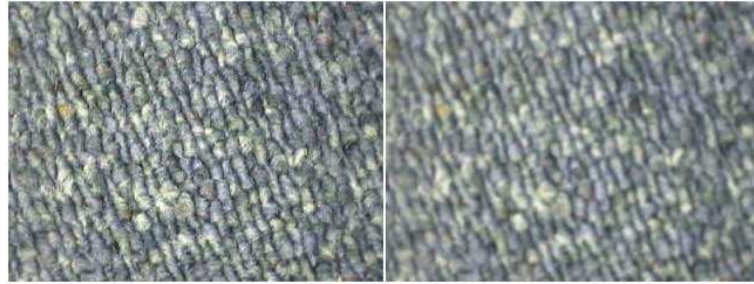


Figure 5.20: Blurring the image to reduce noise levels. Left image shows the original image, while the right image has been spatially blurred using a uniform mask.

A more commonly implemented approach at blurring is the use of a weighted summation of the neighbouring intensities. A sample image before and after the blurring can be seen in figure 5.20. The image on the left is the original, while the right has been passed through a Gaussian filter of size 3 by 3 pixels. The weighting allows the number of neighbouring pixels to be modified when influencing the current pixel of interest, while the individual weights can also be modified depending on how strong the blurring should be. Applying this blurring filter can reduce a large portion of the fluctuating noise, but at the same time, suppresses the intended inter-pixel differences, such as edges or spots in the scene. To avoid combining of unrelated pixel intensities, the neighbouring pixels can be checked with the current pixel for similarity (Peters, 1995). The threshold values that are used are derived from the anticipated noise levels, the standard deviation, for the current pixel intensity. This can be seen in figure 5.21, where the neighbours were blended with equal weighting if the difference in the intensity was less than the standard deviation score for that pixel. The left image shown is the original image and the right image is after the selective blurring filter has been applied.



Figure 5.21: Interpolation based on threshold noise levels. The left image shows the original image, while the right image has been filtered with a selective filter which only blurs if the neighbours are of similar intensity.

Instead of the noise being used as a threshold, the strength used in the blurring can be modified to be proportional to the strength of the noise. This allows for a more controlled use of the blurring to reduce the noise while still retaining some of the

### 5.3.2.1 Spatial filter

intended pixel intensity differences.

As these filters make use of the noise characteristics based on location and intensity, the memory footprint can end up being large, especially when multiple attributes are stored. The use of a lookup tables can typically be faster and more precise than deriving the values dynamically using a closely modelled function. Depending on the size of the table, this can potentially cause time consuming paging operations if the access pattern is not well controlled. Compressing the lookup table is possible, but this requires an decompression phase when the table is used, which can cause spikes in the processing load and cause delays and synchronization issues. It is possible to balance between the dynamic generation and loading of pre-generated values, which is to store the key trend defining values in the lookup table and interpolating between the surrounding points to generate the values in between when required. This idea was also used for the IR sensor reading, and has similarities to the key frames used in video compression techniques. The functions used between the key values can often be made very simple and more efficient than that representing the whole range.

In the approaches discussed above, it is important to consider the processing load when using many neighbouring pixels. Although it is often possible to obtain a more suited model for the current pixel intensity by increasing the number of neighbours to consider, the process must be repeated for all of the pixels, thus has a dramatic effect on the processing load. By noting that the scene does not contain large objects with a predictable intensity structure, using more neighbours does not just decrease the noise, but significantly reduces the fluctuation of the intensities, which can compromise the ability to distinguish between the textures.

It is also worth noting that since the criteria for considering which neighbours to use is generally determined by a simple shaped template, special considerations must be made if the neighbours extends out to an invalid location in the image. For example, when considering the neighbours within 2 pixels of the current pixel, pixels that lie along the border of the image must make sure their neighbour checking algorithm is adjusted appropriately. This can sometimes involve reducing the number of neighbours, not including the overflowing pixels, or even including virtual pixels at the borders so the algorithm does not need changing.

For the purpose of ground texture servo, the typical image being observed often does not contain smooth and continuous regions, as this would not allow significant features to be tracked across frames. This means that a spatial filter can suppress the important information on the transition of intensity even if the occurrence of this is limited with a threshold, thus is not used when observing the ground textures.

### 5.3.2.2 Temporal filter

A different type of neighbour involves the time domain to observe the inter-frame trends from the same pixel location, which can be seen in figure 5.22. Since the fluctuations from the sensor generated noise are centered on the intended intensity, combining multiple samples of the same scene allows for the suppression of the majority of the noise if the scene remains stationary. Other than the lack of motion in both the camera and the scene being the critical requirement for this approach to work, its use is also limited to regions where there are little to no intensity changes.



### 5.3.2.2 Temporal filter

This is because the pixel attempts to portray the transition between the two different colours through aliasing and frequently switches between the two as the dominant colour, even if the camera appears to remain completely stationary.

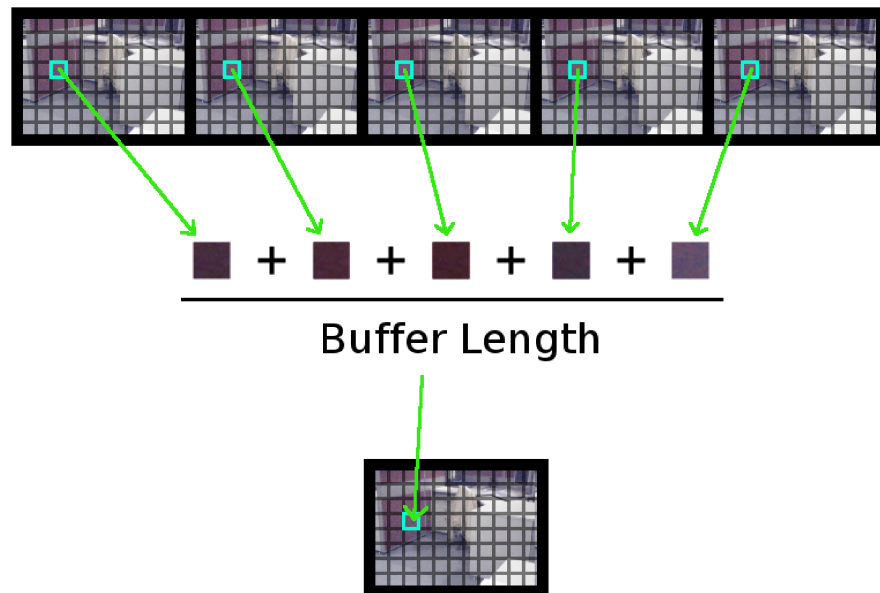


Figure 5.22: Derivation of time based smoothing filters.

The colours are sampled across several frames then averaged to obtain a more consistent intensity.

There are two basic approaches for combining the neighbours. One involves the accumulation of multiple frames then processing the combined image once enough have been stored. The other approach, which is more common, uses a sliding window approach, where the previous frames are stored, usually in a circular buffer, and the weighted sum of the frames within the window is used. Using the accumulation approach, the processed number of frames is reduced by the number of frames stored, while the windowed approach requires a large memory allocation to store the individual frames.

In both these approaches, motions can be quite problematic as the intensity at a particular location can change dramatically. This causes the merged pixel to portray an average intensity which may not reflect any of the colours that were actually present and can also introduce residual objects, much like motion blur. The duration of this artefact depends on the number of frames that are considered in the filter. To avoid this problem, a threshold value can be used to limit the number of frames in the filter. This value can be derived from the noise characteristics to distinguish the change in intensity due to motion or from noise.

By only analysing a single pixel location at a time, the noise characteristic is the only information that can be used to distinguish scene motion from noise. However, in the presence of motion, many pixels belonging to the same object will simultaneously experience motion as a cluster. To exploit this, region based motion constraint or optical flow approaches are used to detect the motion of objects, rather than using single pixel analysis (Barron et al., 1992; Irani et al., 1994; McCarthy & Barnes, 2003). This requires an additional higher level processing load, but can

### 5.3.2.2 Temporal filter

detect the presence of motion quite accurately, thus reducing the artefacts being introduced from the filter. For the ground texture tracking application this idea can be made simpler, since the distance from the camera to the ground remains consistent. This means, with the exception of flat objects that move independently to the floor and perfectly uniform colour, everything in the view moves together. If motion is observed anywhere within the image, the buffered frames can be emptied as the other pixels will also observe motion.

Since the temporal noise reduction filter requires the scene to remain stationary, it is quite limited where it can be applied. When tracking the motion of the ground, the filter may come in use when the robot is stationary on a surface with very limited texture pattern, such as when required to capture a long term landmark. However, since the majority of the robot's execution involves the robot being in motion, this filter would not be applicable for general usage.

### 5.3.3 Hue colour model

So far, all of the algorithms have made use of the red-green-blue (RGB) scale information captured by the camera sensor and have treated them independently when being processed. Many of the existing image processing approaches make use of a grey-scale model instead of using the colours to reduce the processing load and to focus on the shape instead of colour. This is a reasonable approach to make, since many scenes are filled with a variety of colour with no significant links to the semantic information they portray. When the chromatic information is used, the target object is usually of a customised colour to simplify the object identification process or is used as an additional attribute to the object of interest. However, the extra set of relationships provided by the combination of colours distinguishes each pixel and can allow for more reliable correlations to be made. Instead of treating the colours individually, the data can be combined to portray new information (Borzenko et al., 2006, Xu et al., 2006). A simple approach is to map the RGB colour model to an alternate colour model, where the colours are measured in terms of the relative or perceptive values instead of absolute intensities (Ford, 1998).

The absolute intensity is very susceptible to ambient conditions and must often be accompanied by shifting or scaling to account for different environmental conditions. With a relative colour scale, it is more difficult to define a particular set of attributes for a feature, but identifying the presence of an object in the scene is greatly simplified. The idea of inter-pixel relative intensities has been demonstrated in the spatial filters, but it is also possible to use the single pixel to derive inter-colour relative values.

When considering the visual perception mechanism of biological systems, they possess many properties not directly present in cameras. One such property is the high level perception of colour. Although the cone cells on the retina responds to a particular wavelength, much like the behaviour of photo-sensors, the data are combined to form a higher level concept of how the colour is perceived. One set of colour scales which represent the information in similar ways is the hue based colour scale.

A commonly used hue based scale uses three dimensions to define a colour. The

### 5.3.3 Hue colour model

hue defines the relative colour between the red, green and blue, and is represented on a cyclic scale, often visualised like angles in a circle. The richness of the colour is called the saturation and represents the difference between the intensities. The last attribute is can vary depending on the definition of the overall brightness of the colour, and is called luminance in the HSL scale, or value in the HSV scale, which use different points of reference with regards to the saturation. It is worth noting that a grey-scale image can also makes use of the average intensity instead of luminance or value, but generally end up portraying a similar type of information. The formula for mapping from RGB to the HSL scale is as follows.

$$I_{\max} = \max(I_{\text{red}}, I_{\text{green}}, I_{\text{blue}}) \quad (17)$$

$$I_{\min} = \min(I_{\text{red}}, I_{\text{green}}, I_{\text{blue}}) \quad (18)$$

$$\text{Luminance} = (I_{\max} + I_{\min}) / 2 \quad (19)$$

$$\text{Saturation} = (I_{\max} - I_{\min}) / (1 - |(I_{\max} + I_{\min}) - 1|) \quad (20)$$

$$\text{Hue} = \pi / 3 * (I_{\text{green}} - I_{\text{blue}}) / (I_{\max} - I_{\min}), \text{ if } I_{\text{red}} = I_{\max} \quad (21)$$

$$\text{Hue} = \pi / 3 * ((I_{\text{blue}} - I_{\text{red}}) / (I_{\max} - I_{\min}) + 2), \text{ if } I_{\text{green}} = I_{\max} \quad (22)$$

$$\text{Hue} = \pi / 3 * ((I_{\text{red}} - I_{\text{green}}) / (I_{\max} - I_{\min}) + 4), \text{ if } I_{\text{blue}} = I_{\max} \quad (23)$$

where  $I_{\text{red}}$ ,  $I_{\text{green}}$  and  $I_{\text{blue}}$  are the RGB intensities of red, green and blue respectively.

The mapping process from the RGB colour scale to the HSL scale can be carried out dynamically, but it is important to note the differences between the two scales, such as the non-linear mapping, the special cases where hue and saturation is undefined, as well as the cyclic nature of the hue value.

One of the successful criteria for many machine vision techniques is their ability to mimic the human visual perception system. Since we perceive the colours as relative values, many of the objects and the high level constructs we have to describe the object uses the hue like colour model to define the colour attribute. Although the underlying values used in the hue colour scale is still based on an absolute scale, this alternative colour model allows a different perspective that can relate to higher concepts of colour more easily, thus allowing smoother conversion between what the robot sees and what it should see.

Using the hue based colour scale; it is possible to exclude certain aspects of the colour, such as the amount of ambient white light, which can be used to isolate the hue of the object. This allows for the removal of effects like shades and change in the ambient light intensity (Geusebroek et al., 1999). This will be noted by changes in the luminance while the hue and saturation values remain relatively steady.

The shading information that is derived can be used to assist the identification of the object shape, the reflective properties, as well as the characteristics of the light source (Phong, 1975). The removal of these effects greatly simplify the analyses of the surface by being able to cluster the pixels without being affected by colour changes from shadows and lighting conditions, which is especially useful on curved or sloped surfaces.

When being used for the ground texture analysis, the effect of shading is negligible due to the controlled lighting and the absence of obstructions from objects at different height. Although the alternate measure of the texture can be advantageous

### 5.3.3 Hue colour model

in adding confidence to a correlation measure, it does not provide any obvious benefits. This is also due to the undesired clusters it forms on the texture, which removes the important features such as the rough surface textures and grey textured surfaces, not to mention the added computational load.

### 5.3.4 Quantisation blocks

A single image is able to portray an enormous amount of information, but it comes at the expense of requiring a large memory footprint to store the intensity information. When it comes to continuous streams of images, this problem is enhanced, both in the processing load and the transmission load. To overcome the huge bandwidth and memory usage requirements, compression algorithms have been put in place to maintain as much of the original information while removing the redundant and unnecessary component of the images (Gall, 1991).

When inspecting the images acquired from the webcams, it was noted that the images had undergone a moderate amount of compression due to the distinct pattern in the artefacts that were observed. Without the exact specification of the compression technique used by the camera, the specific strategy used was quite difficult to identify. However, the pattern showed the same trends as the compression used for the Moving Pictures Experts Group codec, and subsequently based on the techniques used for the Joint Photographic Experts Group (JPEG) image compression algorithm. Since many applications leave the decoding of the stream to the driver level software, the constructed image must be treated from the artefacts introduced during the compression phase. Figure 5.23 shows a zoomed in view of the captured image where the effects of this compression are very distinctive.

The artefact that was observed is the block formation which is commonly seen in heavily compressed JPEG images. The algorithm attempts to characterise an 8 by 8 pixel block by observing the inner intensity trends using discrete cosine transforms and removing the insignificant components by quantising the values. Depending on the amount of compression used, it causes distinct square patterns around the block and a blurring of the intensity within the block. This block can cause significant bias when considering the inter-pixel transitions, such as edge detection algorithms.

Observations of the captured image showed the distinctive block formation, but with a dimension of 4 by 4 pixels and a weaker block formation of 2 by 2 pixels inside. This was caused by the size of the block being considered and from the weighting pattern used in the quantization matrix. To reduce the distinct blocks from biasing the various algorithms, the pixels surrounding the border of the square were blended in with the neighbouring pixels. By observing this trend in the transitions, an interpolation mask can be derived to focus on the appropriate portions of the image.

Since the pixels have been influenced by all of the pixels within the block while some of the information has been lost, reconstruction of the original intensity becomes a very difficult and time consuming process. By using a lower capture resolution, it is possible to reduce the number of these blocks forming, but reduces the precision available for the camera. By increasing the resolution, it will introduce more blocks since the dimensions of the blocks remain the same. However, it is able to capture the details of the environment and allow for a better smoothing algorithm.

### 5.3.4 Quantisation blocks

The evaluation of the appropriate weighting is difficult to achieve due to the inability to capture the noise-less version of the same scene. When using the different resolutions, the lower resolution image was too imprecise and the intensities did not correlate well due to the large amount of blending which had already occurred. Using the higher resolution showed an interesting behaviour, where the amount of compression was increased, making the blurring effect within the block and the difference at the borders of the blocks much stronger. This effect was caused by the attempt to maintain similar throughput with the larger volume of data.



Figure 5.23: Block formation from compression.

The codec groups the 4 x 4 squares and blends them, creating a distinctive border between these squares.

Since the significant issue with the block formation was with the inter-block boundaries, the weights in the interpolation was adjusted accordingly to promote smoothing while retaining most of the original appearance. Several weight values were tested for the merging of the bordering pixels and were judged manually on the effective and accurate noise removal. Figure 5.24 shows the original image and a typical Gaussian blur algorithm being applied, while figure 5.25 shows the various weights and the resulting image from the customised filter.



Figure 5.24: Original image and the result of applying a Gaussian blur filter.

Left image is the original, while the right image has been blurred using a Gaussian filter.

### 5.3.4 Quantisation blocks

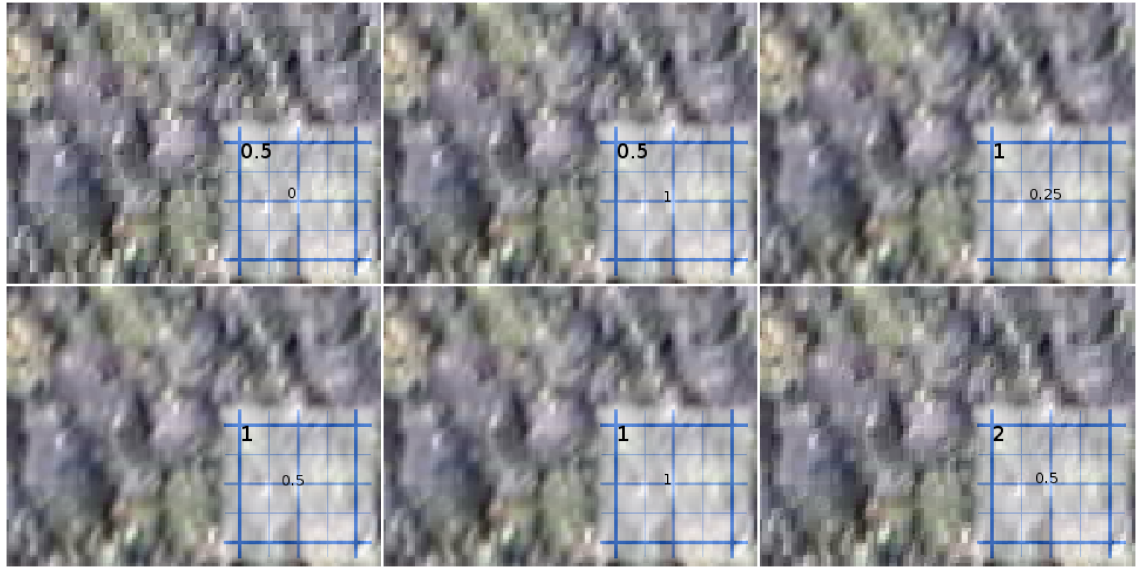


Figure 5.25: Customised block removal weights and the resulting images. The weights used are shown in the small grid at the bottom right of each image. The top left number is the weight used between the 4 x 4 squares, while the number in the middle is used between the internal 2 x 2 squares.

## 5.4 Summary

The use of a webcam has allowed easy integration of visual processing capabilities on mobile robot platforms. The low quality and the lack of dynamic control over camera attributes can be compensated for by carefully analysing the characteristics of the camera and the environment it is operating under. These characteristics can then be applied to the camera settings, configurations, and the image filters to take advantage of any known constraints to improve its effectiveness and to restore the image from various artefacts that are introduced by the device.

Many conditions and configuration issues of the webcam were investigated, as well as image filter algorithms to target specific artefacts to be removed. In both cases, the settings and the use depend greatly on the specific application and the available resources. For the task of ground texture servo, the camera settings have been well defined due to the constraints of the platform. The image filter algorithms, however, depends greatly on the processing capability of the system and the required level of accuracy.

The majority of the noise encountered by the camera is negligible for the ground texture servo task. This is due to the characteristics of the ground textures, which often contains locally unique surface structures and textures when viewed from a close distance. However, there is scope for the filters that were discussed to be applied for other webcam applications where image restoration is more critical.

Due to the limited resources, the selection of the filters to be used had to be done carefully and to not waste the precious processing cycles and memory usage on a filter that did not make significant differences. The various intensity based noise characteristics required an enormous memory footprint, so they were approximated

## 5.4 Summary

to a polynomial function instead. The spatial and temporal filters which made use of these characteristics showed some signs of noise being removed, but consumed a reasonable processing time. Since the ground textures could still be distinguished without the filters, these neighbourhood based approaches were not implemented as part of the ground texture tracking system.

Since the lighting configuration is known for this application, the radial shifting characteristics allowed for a significant improvement in evening out the image intensities. The typical operating surfaces for the mobile robot are the carpet and vinyl flooring, where the reflective properties differ significantly. This meant controlling the amount of adjustment involved the use of the average intensity difference between the center portion of the image and the outer edges of the image. The current implementation performs this check every 1000 frames to ease the processing load, which equates to approximately once every 30 seconds.

The most significant filter to be included is the removal of the blocks caused by the image compression algorithms. Although the weighting values have not been fully explored, the filter has shown significant improvement in the image appearance by removing the artefacts without introducing too much artefacts of its own from the blurring.

The configuration process and the filters that have been defined form the primary step of processing image streams from webcams. The characteristics of the filters can be used to identify which situations they can be applied in to improve the analysis processes on the images.

## Chapter 6 – Feature Tracking

Successfully navigating around an environment feels like a trivial task for many biological systems, but defining the complete process from a series of primitive instructions is a tremendously difficult task. Although many researches are influenced by the workings of biological systems, acknowledging the differences can often lead to a more efficient and precise system (Mondada & Franzi, 1993).

The use of cameras for navigation attempts to mimic the eyes, but this requires a significant amount of processing capability, adjustable sensitivity, as well as memory capacity in recognising similarity between objects. An electronic system is able to perform consistent and regular actions with high degrees of precision, which can be made use of to assist this task.

Visual odometry approaches involve identifying the location or the change in the location of features and landmarks in the environment and in turn, deriving the location of itself. This requires three major phases, which consist of identifying candidate features within the captured image, tracking their motions by correlating with the stored features from a different perspective, and finally establishing the location of the feature and the mobile robot (Krootjohn, 2007; Lucas & Kanade, 1981).

The area of feature tracking algorithms is well studied and already consists of many fundamental techniques (Ritter & Wilson, 1996; Shi & Tomasi, 1994) and related algorithms (Fleet & Langley, 1995; Isard & Blake, 1998) for achieving reliable and efficient tracking of points of interest. However, many projects customise these existing algorithms by analysing the problem from a particular perspective. This constrains the parameters involved in the algorithms, as well as including other constraints to suit the particular task.

The proposed approach involves the use of a downward pointing camera to measure the displacement of the ground by identifying and tracking patterns across frames. Using a simple triangulation technique, the motion of the ground can be translated to the motion of the robot since the technique assumes that the distance between the camera and the ground remains constant. The instantaneous displacement can then be accumulated to form 2D pose information with respect to the starting point.

The camera sensor, which is sometimes categorised as a directional sensor, is able to associate a direction of the incoming intensity through the position of the photo-sensor, thus often measures its precision characteristics in terms of degrees. As the distance between the camera and the observed object increases, the precision of the observed object decreases since the arc length is proportional to the distance. However, with the proposed camera configuration, this error rate can be set quite low due to the small distance to the ground, and more importantly, stays constant.

Before the images are analysed, they are filtered using several algorithms described in chapter 5. By allowing the ground texture to move a significant distance



## Chapter 6 – Feature Tracking

around the image, the approach runs the risk of encountering repeated texture patterns, especially on smooth surfaces. This means the viewing area and the maximum speed of the robot should be reduced, which allowed the outer edges of the captured image to be discarded, as per the lens distortion correction. Two more artefact removal filters were implemented, which were the radial shift adjustment filter and the block formation restoration filter.

The change in the sequence of the filters did not show any obvious visible effects, thus the radial shift was applied first to discourage too much shifting from the original intensity. Since the access patterns of the pixels are different, the two filters could not be merged together. However, the block restoration process benefited from caching some of the merging between the pixels, as each merge is used twice for the two pixels that are on the border. This can be seen in algorithm 6.1 below. Note that the implementation shown below illustrates the caching approach and is not the optimised algorithm. For this particular algorithm, the sequence of positions is well structured, thus the loops and the conditional statements can be unwrapped to improve the performance.

```
function RemoveQuantisationBlock(image, weight_strong,
weight_weak):
    set v_cache_array[image.width] = { 0... }
    set restored_image[image.width, image.height, 3]
    for row in 0 to image.height:
        set row_mod = row % 4
        set h_cache = 0
        for col in 0 to image.width:
            for rgb in {red, green, blue}:
                set value = 4 * image[col, row, rgb]
                set weight = 4
                switch col % 4:
                    case 3:
                        if col == image.width-1:
                            set h_cache = 0
                            break
                        else:
                            set h_cache = image[col+1, row, rgb] *
                                weight_strong
                    case 0:
                        set value = value + h_cache
                        set weight = weight + weight_strong
                        break
                    case 1:
                        set h_cache = image[col+1, row, rgb] *
                            weight_weak
                    case 2:
                        set value = value + h_cache
                        set weight = weight + weight_weak
                        break
                switch row_mod:
                    case 3:
                        if row != image.height-1:
                            set v_cache_array[col] = image[col, row+1,
                                rgb] * weight_strong
                    case 0:
                        set value = value + v_cache_array[col]
                        set weight = weight + weight_strong
```

## Chapter 6 – Feature Tracking

```
        break
    case 1:
        set v_cache_array[col] = image[col+1, row,
            rgb] * weight_weak
    case 2:
        set value = value + v_cache_array[col]
        set weight = weight + weight_weak
        break
    set restored_image[col, row, rgb] = value / weight
return restored_image
```

Algorithm 6.1: Algorithm for quantisation block filter.

When integrating the three algorithms, the changes in the speed and memory consumption were monitored to allow more capacity in the image analysis phase.

The different phases of visual odometry are investigated in more detail to identify the set of criteria to be used for the localisation technique. Considerations such as the effectiveness, accuracy, and processing requirements were made when evaluating the various approaches.

### 6.1 Feature detection

The first of the three phases is one of the most well studied area in image processing as it forms the foundation for most high level processing tasks. The difficulty of identifying and extracting the relevant information from a grid of intensity measures can be as complex as desired, depending on the available equipment, requirements, the reliability of a priori information, and the constraints that can be placed to assist in simplifying the task.

Although the domain is known, applying too many constraints yields a very inflexible algorithm that is severely limited in where it can be applied. To avoid this, only the generic and crucial domain knowledge is applied, such as the repetitiveness of the textures, the lack of variety in the intensity levels and patterns within a single frame, the constant elevation of the camera, lack of depth queues, some availability of motor commands, and limited motion constraints from the wheel configuration (Draper et al., 1996).

In addition to the above, an extra constraint is introduced which assumes that no rotations can occur between the captured frames. As the majority of correlation algorithms tends to exhaustively search the available space or require very sophisticated dynamic programming algorithms to prune the search space for the best correlation, this constraint allows for a significant reduction in the search space, as it narrows down the possible dimensions down to 2. This is because the transformations are prohibited due to the fixed elevation (Lowe, 1999; Matas et al., 2000).

The constraint mentioned above is made possible due to the fast frame rate, the pace of the robot, and controlled motions. The short interval between the capturing of the frames allows for the transformation of the feature to remain very small, thus almost eliminating the rotation that can occur. By decreasing the operational speed of the robot, it can also reduce the rotation in a similar way to increasing the frame rate. The difference between the two strategies includes catering for non-self powered

## 6.1 Feature detection

motion when the frame rate is modified, and the reduction in the operational speed of the robot does not cause the increase in the processing load.

When considering just the motion of the webcam, successfully eliminating transformations and partial translations requires the precise control of all motions of the camera. Since this is an unreasonable constraint to uphold, the camera should be placed to minimise the blurring of the pixels. This can be achieved by placing the camera away from the rotational point, as the amount of transformations that occur is greater towards the pivot point. For a differential drive system, the motions of the robot can be restricted while it remains in the normal mode of operation. That is, the robot is propelled by the self induced forces. This information, together with the wheel arrangement, can be used to place the camera away from the rotational axis of the wheels. When rotations do occur, it will consist of the combination of translation and small amount of rotation. The camera placement will be discussed in further details in chapter 7.

### 6.1.1 Lifetime

Another important consideration to make before the image is analysed is the effective lifetime of the features. In most feature detection algorithms, the process involves identifying an object, or attributes of the object, that allows it to be distinguishable from multiple perspectives (Paletta et al., 2005). This implies that the feature is to be tracked for a reasonable period of time as it moves around within, and possibly even out of, the view. The process of identifying these features requires non-morphological attributes to be extracted and also requires the presence of a reasonable candidate to be present in the view.

Since the typical view observed by the ground pointing camera consists of repetitive texture patterns, low contrast, and the limited viewing area, successfully identifying and tracking a feature is made considerably more difficult. These conditions mean that for a real time system, the feature identification process must be very rapid in identifying its uniqueness and be effective enough to be able to track its motion in subsequent frames (Davison, 2003). The major contributor to the limited availability of the features is the small elevation of the camera and the operational speed of the robot, which controls the rate of movement of the ground textures. Although these can be adjusted to reduce the texture motion, the increase in the height reduces the precision, while slowing of the robot can conflict with the speed requirements of the robot. This means the feature may only be observed in the immediately subsequent frame and fall outside the viewing area later. This can potentially cause issues like incorrect tracking occurring when a non-unique feature is selected, especially as no confirmation of the tracking can be provided.

Although the limited lifetime results in frequent re-computation of the feature, it has several positive side effects, such as reduced morphological transformation of the feature and the reduction in the search space as the feature will only need to be tracked once from a confirmed position in the image. This also means if a feature is badly chosen and corrupts the localisation process, the effect will only contribute to a very small amount of error as a new feature will be used in the subsequent frame.

Since the feature does not need to consider morphological transformations, the

### 6.1.1 Lifetime

attributes that require consideration only includes the amount of longitudinal and latitudinal shifts, as well as the intensity information, which may be altered by the change in the ambient light or from sub-pixel motion. These attributes can be considered by specifying the shape and size of the search area, as well as the intensity based information which includes the raw intensity and also some derived patterns from the pixel intensity arrangement, such as intensity transition strengths.

### 6.1.2 Score

One of the key criteria in selecting a feature is the ability to identify a region that remains unique and identifiable even after some degrees of transformation (Peters & Strickland, 1990). An approach which only uses the intensity values is very simple to implement, but does not allow reliable performance when there are changes in the ambient light, rotations, or sub-pixel motion. Due to the constraints placed by the system, which limits the changes in the ambient light and the camera rotation, the intensity based approach eliminates two of the issues.

By using the intensity values, the inter-pixel information, and the locations of the pixels, it is possible to derive many scores which can be used to assign unique attributes to the feature. The correlation between just the intensity values is the most frequently used approach due to the simplicity, as it simply requires the measure of the sum of the difference between the intensities across the region used for the feature, which is often accompanied with limited change in the location to manage the search space (Huttenlocher et al., 1993).

Using the inter-pixel information, such as edges, which are the differences between the intensities (Harris & Stephens, 1988; Hildreth, 1985; Liao et al., 1997; Torre & Poggio, 1986; Ziou & Tabbone, 1997), it is able to portray the relative information in case global intensity change occurs from events such as shades or the room light changes. Using this measure by itself does not allow for the absolute reference point, thus can potentially correlate with a significantly different texture, but with a similar arrangement in the pixel intensity change. This approach can also be made use of to identify higher level constructs to the intensity pattern, such as edges and corners that are not constrained by the current perspective (Smith & Brady, 1997). However, these techniques often require a larger viewing area to support the findings of the constructs.

One other frequently used approach is a template based approach, where a particular intensity or frequency pattern is determined beforehand by specific arrangement of the intensity or simply by setting the valid bounds. The template is then used to find the closest matching candidate as the feature. Although determining a good candidate can be simplified, this approach requires a priori knowledge on the expected features and potentially very sophisticated and time consuming pre-processing of information to convert the data to fit the template, together with dynamic adjustments to account for new surfaces.

Although the feature identification process primarily focuses on the unique region, it is also important to capture the surrounding information to identify the context of the feature. The increased area of the feature improves the reliability as it introduces more attributes to the correlation process. This also prevents the feature

### 6.1.2 Score

from being ineffective when the main unique portion is corrupted or modified through blurring and sub-pixel motion.

To identify the effectiveness of a feature, a scoring scheme was devised to compare and rank the feature candidates. The uniqueness of a feature is influenced by the attributes discussed above, thus considerations were made to determine which approaches and attributes would best suit the criteria for a good feature. The processing time required to assign the score is also included, as this plays a crucial role in real time operation where new features are constantly required.

The first and the simplest approach makes use of the difference between the average intensity, or the standard deviation score, to identify a region showing the most fluctuation. The score is derived for three different averages, where the first only considers the region included for the feature; the second considers the pixels within the search area for the candidates, while the last considered the whole image. The algorithm for the whole view can be seen in algorithm 6.2 below. The features which were identified through this process depend heavily on the current view and do not include a uniqueness value into the score evaluation. In terms of the processing load, this approach requires two parses to determine the average intensity then accumulating the difference for each pixel within the feature.

```
function FindStdDevCandidate(image, candidate, feature):
    set sum[3] = {0, 0, 0}
    for row in 0 to candidate.height + feature.height:
        for col in 0 to candidate.width + feature.width:
            for rgb in {red, green, blue}:
                set sum[rgb] = sum[rgb] + image[candidate.x + col,
                    candidate.y + row, rgb]
    set average[3] = {0, 0, 0}
    for rgb in {red, green, blue}:
        set average[rgb] = sum[rgb] / ((candidate.height +
            feature.height) *
            (candidate.width + feature.width))
    for row in 0 to candidate.height:
        for col in 0 to candidate.width:
            set score = 0
            for v in 0 to feature.height:
                for v in 0 to feature.width:
                    for rgb in {red, green, blue}:
                        set score = score + abs(image[candidate.x +
                            col + v, candidate.y + row + v, rgb] -
                            average[rgb])
            if score > feature.score:
                set feature.score = score
                set feature.x = candidate.x + col
                set feature.y = candidate.y + row
    return feature
```

Algorithm 6.2: Feature score based on the difference between the average score.

Using the difference between the neighbouring intensities, the amount of fluctuation in the differences was accumulated as the score for ranking the feature candidates. This alternative approach requires the evaluation of both the horizontal and vertical intensity difference for each pixel, as a weak difference can lead to ambiguity in that direction and can cause the features to slide along the axis. This

## 6.1.2 Score

approach allows for a more consistent measure in the presence of ambient light changes.

Since this algorithm is interested in accumulating the overall fluctuation of the intensity, the direction of the intensity change and the per-pixel information is not important. This means that the magnitude of the difference in the intensity can be accumulated for all the inter-pixel transition points instead of iterating over each pixel. Algorithm 6.3 illustrates this process, while the idea of traversing over the transition point between the pixels is discussed later.

```
function FindSumDifferenceCandidate(image, candidate, feature):
    for row in 0 to candidate.height:
        for col in 0 to candidate.width:
            set score = 0
            for rgb in {red, green, blue}:
                for v in row to feature.height + row:
                    for h in col to feature.width + col:
                        set score = score + abs(image[candidate.x +
                            h, candidate.y + v, rgb] - image[candidate.x +
                            h - 1, candidate.y + v, rgb]) +
                            abs(image[candidate.x + h, candidate.y + v,
                                rgb] - image[candidate.x + h, candidate.y +
                                v - 1, rgb])
                        set score = score + abs(image[candidate.x +
                            feature.width, candidate.y + v, rgb] -
                            image[candidate.x + col + feature.width - 1,
                                candidate.y + v, rgb])
                    for h in col to feature.width + col:
                        set score = score + abs(image[candidate.x + h,
                            candidate.y + row + feature.height, rgb] -
                            image[candidate.x + h, candidate.y + row +
                                feature.height - 1, rgb])
            if score > feature.score:
                set feature.score = score
                set feature.x = candidate.x + col
                set feature.y = candidate.y + row
    return feature
```

Algorithm 6.3: Feature score based on the sum of the fluctuation in neighbouring pixel intensities.

### 6.1.2.1 Edge map

The process of searching for the ideal feature typically involves moving a viewing window and evaluating the score of the pixels bound by the window until the search space has been exhaustively searched or it has been deemed that no better feature can be found. This means most of the regions within the view will be visited multiple times as it contributes to the feature from different starting location of the window. By storing the difference information for the whole view, it can avoid the re-processing the transition information calculated between the pixel pairs.

This idea is similar to that used in the quantisation block algorithm, where the evaluated intensity information is maintained and used in the subsequent iteration. This allows the number of evaluations to be reduced to almost half, since the buffered values are only encountered twice. The extra memory requirement for this is

### 6.1.2.1 Edge map

quite small, which only requires an array to store the vertical transition, as well as a single buffer to store the latest horizontal transition value.

The difference between the intensity values are frequently used for many algorithms, hence it can be beneficial to implement an efficient way to access this information. Depending on the type of edges to be considered, such as the direction, the number of neighbours to consider, and the access patterns, the storage location of the intensity differences can be customised configured.

Many applications which make use of the intensity changes often neglect or approximate the aliasing effect and use the pixel coordinates to represent the overall change in intensity experienced at that point. This is typically done by mapping the edge scores back to the central point of the pixel, which can cause some edge information to interfere with each other and can also increase the misalignment between the edge location and the actual edge in the scene. By isolating the intensity changes at different locations and storing them separately, it can maintain a more precise representation of the edge information. Due to the extra level of redundancy, this requires a larger memory footprint. As the edge values are relative, it is possible to reduce the memory footprint by allowing iterations to derive the value at a particular point. However, this introduces more processing load which defeats the purpose of pre-calculating the transition information. Figure 6.1 illustrates the positioning of the edge map.

The current implementation only makes use of the intensity difference values in the immediate neighbours that are accessed in a simple sequential scan pattern. Although this requires one of the dimensions to jump back as the other reaches the end, the dimensions of the image is too small to cause dramatic problems with paging. Instead of arranging the map so that it suits the particular access pattern involved in the difference approach, offsets and jumps are used to allow other algorithms to make use of the edge map without the hassle of arranging a complex lookup pattern to access the values.

Calculating and storing the difference score beforehand allows for some speed ups, but this can be further optimized by combining the portions that are moved in and out of the window area to quicken the process of evaluating the new score when the window shifts. This accumulation of the score can be done at the same time as the difference evaluation, or on the fly, such as when the newly included pixels are accumulated individually and stored to be used when the group of pixels are removed from the window. This process is illustrated visually in figure 6 2 below, where case 1 is the initial process, case 2 is when the window is shifted horizontally, and case 3 is when the stored score at the start of the row is brought back.

### 6.1.2.1 Edge map

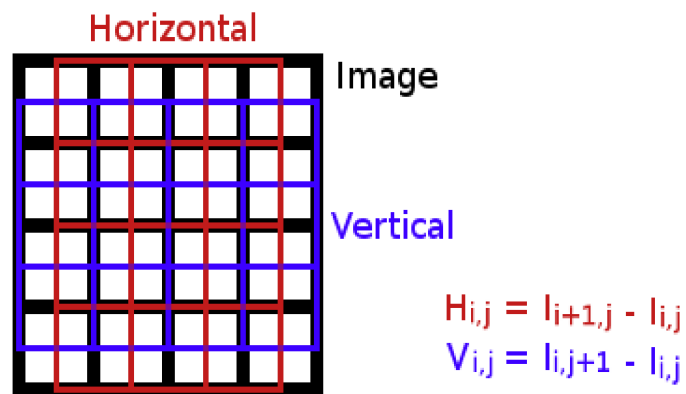


Figure 6.1: Illustration of the edge map. The red squares represent the intensity difference between the horizontal neighbours, while the blue squares represent the intensity difference between the vertical neighbours.

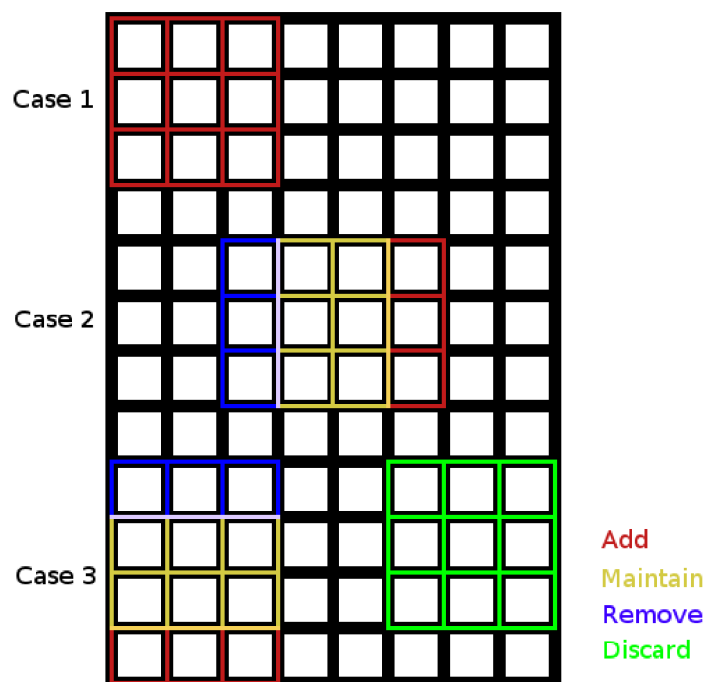


Figure 6.2: Evaluation of the score by applying the difference in scores as the window moves.

Case 1 shows the initial point, where all cells must be evaluated. Case 2 shows the usual transition where the left most column is removed and the cells to the right of the previous region is added. Case 3 shows the wrapping between multiple rows by reverting back to the state at the beginning of the row and applying a vertical version of that seen in case 2. The scores from the green cells are thus discarded.



### 6.1.2.1 Edge map

This difference approach of the window, as well as the accumulation of the portions being changed, can lead to decreased processing time due to smaller computational cost in adjusting the current score rather than evaluating a new score by scanning through the entire window. Using the standard sequential approach, the number of accesses to the score of each pixel is:

$$\text{Width}_{\text{feature}} * \text{Height}_{\text{feature}} * \text{Width}_{\text{candidate}} * \text{Height}_{\text{candidate}} \quad (24)$$

Using the proposed approach, this can be reduced to:

$$\text{Width}_{\text{feature}} * \text{Height}_{\text{feature}} + 2 * \text{Height}_{\text{candidate}} * (\text{Width}_{\text{candidate}} * \text{Height}_{\text{feature}} + \text{Width}_{\text{feature}}) \quad (25)$$

The above can be reduced even further by rotating the image or traversing the candidates vertically. Since the values at each pixel are visited at least twice, it is important to cache this value or derive the value before the accumulation of the score. Algorithm 6.4 below shows the score evaluation process by using the difference in the scores, while table 6.1 summarises the effects of the implementation compared to a simple approach of summation of the intensity and the difference in the intensity.

```
function FindMaximumSumCandidate(image, candidate, feature):
    set score = 0
    for row in 0 to feature.height:
        for col in 0 to feature.width:
            set score = score + GetScore(image, candidate.x + col,
                candidate.y + row)
    for row in 0 to candidate.height:
        if score > feature.score:
            set feature.score = score
            set feature.x = candidate.x
            set feature.y = candidate.y + row
        set row_score = score
        for col in 0 to candidate.width - 1:
            for v in row to feature.height + row:
                set score = score + GetScore(image, candidate.x +
                    feature.width + col, candidate.y + v) -
                    GetScore(image, candidate.x + col, candidate.y +
                    v)
            if score > feature.score:
                set feature.score = score
                set feature.x = candidate.x + col
                set feature.y = candidate.y + row
        score = row_score
    for h in 0 to feature.width:
        set score = score + GetScore(image, candidate.x + h,
            candidate.y + feature.height + row) - GetScore(image,
            candidate.x + h, candidate.y + row)
    return feature
```

Algorithm 6.4: Evaluation of score using the difference as the window shifts.

Table 6.1: Execution time of implementing the difference algorithm.

	Sequential algorithm (ms)	Difference algorithm (ms)
$\Sigma$ Intensity	2.48	1.29
$\Sigma\Delta$ Intensity	3.21	2.04

### 6.1.2.2 Uniqueness

### 6.1.2.2 Uniqueness

The attributes used above focuses primarily on the interestingness of the feature with respect to the current view or against some range of possible intensity arrangements. Although this attribute contributes to the effectiveness of the feature, a more important measure to consider is the uniqueness of the feature within the current and the subsequent or the previous frame, depending on when the tracking is done. The uniqueness is a measure of how distinctive the feature is, thus requires the comparison to the other feature candidates.

The brute force approach of comparing the feature with every single candidate for the most uniqueness requires  $O(n^2)$  comparison between the features, where  $n$  is the number of feature candidates. This search space can be reduced with sophisticated algorithms such as beam search to prune away the bad candidates early on or using A\* like algorithms to rank and prioritise the processing of good candidates first. For a search space with differences in the scores, the overheads in implementing the algorithms often outweighs the gain in speed. This is also true when is a low number of candidates, thus a simpler algorithm is more applicable for real-time applications by restricting the candidates and making assumptions about the features it will observe.

As the candidates are exhaustively searched, the scores that are accumulated while portions of the feature are examined can be compared against a threshold value for early termination. This simple elimination can be introduced by comparing the candidates one at a time and maintaining the best score so far. Since the scores can be made to be accumulative, the current score should always be better or equal to the best score achieved so far. If the score becomes worse than the current best candidate, the search can be terminated for that candidate as the end score will always be worse than the best one so far.

Another approach is to compare the difference against other candidates to identify an outlier. The efficiency of this approach greatly depends on the algorithm used to assign the scores which requires a large range of values and the actual presence of outliers from the clusters that form amongst the scores. Using a single score compresses the clustering problem into a simple distance comparison.

This approach can also make use of the threshold value for early termination, as the desired feature will have the largest minimum distance between the scores. If an evaluated score between a candidate pair is less than the current threshold value, both candidates can be discarded as not being the most distinctive score. It is also possible to make use of other simple clustering algorithms, such as an uniform grid spaced density approach using bucket sorting, but this does not guarantee the selection of the most distinctive feature as it finds the local maxima, thus requires multiple parses to re-cluster the scores to identify the most distinctive feature.

By observing the trends between multiple frames, certain behaviour, such as the presence of flickering lights and repetitive patterns in the texture can be observed to influence the uniqueness scores. The approach can also be used to determine the general flow of the ground using techniques such as optical flow that can assist in the disambiguation of similar patterns. The identification and derivation of these trends often requires significant amount of resources or strict constraints on the

### 6.1.2.2 Uniqueness

environment the approach can be used. The typical view observed by the cameras consists of repetitive and non-distinctive intensity patterns, which make the process of identifying the trends even more difficult. There are no guarantees in the presence of trends, while general optical flow approaches often does not work in the presence of non-distinctive textures due to the small pixel area being tracked. Although these approaches are not included when determining the best feature candidate to use, the idea is used when tracking the feature, as described in detail later. This is due to the extra constraints that can be enforced as the desired target is known.

The ability to identify the most unique feature is desirable, but it is not a necessity as long as the feature can still be tracked using other constraints. This implies that the resource usage and the validity of the uniqueness score must be balanced to best utilise the processing capability.

The approaches above were compared using a constant rectangular feature size of 16 by 16 pixels against 64 candidates by sequentially shifting the feature window in a 8 by 8 square. To compare the result against naïve algorithms, several simple algorithms were introduced, which included the selecting of the brightest and darkest features, the most average feature, and also a randomly selected feature. The process involved averaging the various attributes across 500 frames, which consisted of the average time taken, the utilisation of the range of values possible which is calculated by the difference in the minimum and maximum observed value divided by the range, and also the uniqueness score, which is the percentage rank of uniqueness determined by comparing the distance in the scores. The tests were carried out on two common ground texture types, a carpet floor and vinyl flooring, which contained less distinctive texture patterns than on carpet. Table 6.2 below summarises the results from the experiment.

Table 6.2: Comparison of feature selection algorithms.

	Carpet			Vinyl		
	Time (ms)	Utilisation (%)	Uniqueness (%)	Time (ms)	Utilisation (%)	Uniqueness (%)
$\max(\Sigma I)$	2.33	3.21	91.23	2.27	0.44	97.45
$\min(\Sigma I)$	2.31	3.26	73.23	2.41	0.41	89.06
$\text{mid}(\Sigma I)$	2.34	3.22	39.03	2.29	0.46	7.83
$\text{rand}(\Sigma I)$	2.38	3.31	49.57	2.33	0.43	51.46
$\max(\Sigma  I_{x,y} - I_{\text{ave}}(\text{all}) )$	3.9	1.32	99.67	3.63	0.61	97.54
$\max(\Sigma  I_{x,y} - I_{\text{ave}}(\text{view}) )$	3.81	1.22	96.66	2.61	1.48	95.55
$\max(\Sigma  I_{x,y} - I_{\text{ave}}(\text{feature}) )$	4.09	1.44	97.1	3.01	0.58	88.05
$\max(\Sigma  I_{x,y} - I_{x+1,y}  +  I_{x,y} - I_{x,y+1} )$	3.37	1.26	65.58	3.85	0.56	37.12
$\max(\Sigma  I_{x,y} - I_{x+1,y}  *  I_{x,y} - I_{x,y+1} )$	3.43	0.19	57.25	4.43	0.01	93.74

The low utilization scores in all of the algorithms is due to the limited range of intensities that are available within the small viewing area. This was to be expected due to the repetitive nature of the ground textures. This means the algorithms which are based on the relative intensities are affected even more due to the similarity in the intensity and the blending which occurs from the aliasing.

### 6.1.2.2 Uniqueness

The results above show that the naïve approaches differ greatly. However, it was noted that the majority of the best utilised feature either had the highest or lowest score. This means the utilisation score can be further improved by tracking the top and bottom two scores and selecting the one with the greater difference.

Using the averaging algorithm indicated high level of uniqueness, but showed slower performance. Note that the algorithms were implemented without using the difference algorithm introduced earlier, thus can potentially be sped up to improve the performance. The edge score based approaches, on the other hand, did not perform well due to the repetitive nature of the texture patterns. The lower utilisation for the product of the edge score is due to the much larger range and the non-linear distribution of scores.

The low uniqueness score means it would be more difficult to identify the feature, as there are more candidates with similar scores. However, it is important to note that although the feature selection process may base the criteria on one attribute, the feature tracking algorithm used to correlate between features does not have to rely on the same attribute. The uniqueness is simply one measure to rank the feature candidate for selection.

One of the major issues with the edge based approaches is the effect of sub-pixel motion, which can significantly modify the scores. The lack of absolute information can also reduce its effectiveness when shaped patterns are repeated. Although this is also problematic when the intensity information is used, the controlled lighting and the ability to interpolate absolute values to anticipate intensity changes from sub-pixel motion means it is more attractive for this application. Since the feature must be tracked around the view, the use of the viewing area average allows better portrayal of the feature's effectiveness, thus is implemented as the current feature selection algorithm.

### 6.1.3 Shape and size

When specifying the characteristics of the feature, attributes such as the size and the shape contribute greatly in controlling the reliability, processing requirements, as well as the ability to make use of any a priori knowledge about the typical types of textures it will observe. Using a small feature would allow faster processing, but suffers from the reduced variety in the captured intensity, which causes non-distinctive features to be selected. A larger feature, on the other hand, can allow more distinct features to be selected due to the additional constraint from the extra pixels. However, this leads to the increased processing requirement and the additional memory consumption in maintaining the intensity and any derived values.

Since the feature involves analysing a group of pixels, a consideration into how the pixels should be arranged is required to make the most of the available pixels for the feature selection process. The different arrangement formed by the pixels can allow different styled features to be captured since it can cater for certain trends to be captured more effectively than others. The use of a square or rectangular shape is the most common approach due to the sequential access pattern and the simplicity in the implementation. This is also due to the arrangement of the pixels in most images where the pixels are of uniform sized squares, arranged densely one after the other.

### 6.1.3 Shape and size

Deviation from this shape can include the change to other primitive shapes, such as circles to focus attention to a particular point rather than a region, or a line if certain motions can be constrained to only have one unknown dimension. This choice depends on a priori information about the ground textures and the orientation considerations which can affect the placement and the number of pixels required at a certain location.

The use of the solid rectangular shape also makes use of some domain knowledge about the frequency or the distinctive textures in one direction to another. As well as the efficient indexing, the density can also assist in the case of when motion blur occurs by capturing the adjacent information to compensate for the interpolation in the intensity. The orientation and frequency of the motion blur plays an important role in the above argument, which is constrained through the original assumption that the feature will primarily translate in one axis, and the effect of the blurring will be limited due to the short exposure time.

When observing the trends of the scores on some surfaces, it was noted that some portions of the image did not make much of a contribution to the scores, especially when using the edge based scoring techniques. Instead of being constrained to primitive shapes, alternative shapes were designed to take advantage of this observation to determine the effectiveness of modifying the shape to suit the texture.

This has led to the formation of a straight edged donut shape where the central square portion of the feature is removed. This is due to the structure of the carpet floor, where the bundle of threads that make up the grooves are present. The regions between these bundles show large differences in the intensity, whereas the bundled portion itself does not show any interesting transitions for edge based scores to make use of. By skipping over these regions through aligning the hole with the bundled portion, the processing load should be reduced without hindering the effectiveness of scores. The purpose of this shape was to measure the effectiveness of a customised shape based on the observed texture, which could be determined dynamically.

When implementing the donut shaped approach, an extra requirement is introduced to make sure the feature candidates include the view with the optimally aligned position. This can potentially increase the number of candidates depending on the interval and the size of the unattractive region.

The last shape to be investigated was a scattered grid formation with uniform spacing, intended to maintain reasonable efficiency through regular intervals while increasing the coverage at the cost of the neighbouring details surrounding a pixel. This drawback means interpolation of neighbours will not be possible when sub-pixel motion occurs.

By extending the idea of configuring the feature shape to suit the environment, an approach involving a dynamic analysis of the surface texture pattern is required. The use of this approach can potentially provide the best range of scores for a fixed number of pixels. However, like other dynamic approaches, the periodical evaluation of the current state and configuration would cause significant overheads, as well as relying on the presence of predictable patterns it can exploit. This is due to the rapid changes in the texture caused by the small viewing area and the frequent motion of the robot. Instead, the manually configured shape and sizes will indicate the validity and effectiveness of the shape and configuration attributes to suit the environment.

### 6.1.3 Shape and size

They will also allow the switching between these configurations for particular textures if desired, as the operating surfaces of the robot will be limited in the majority of situations.

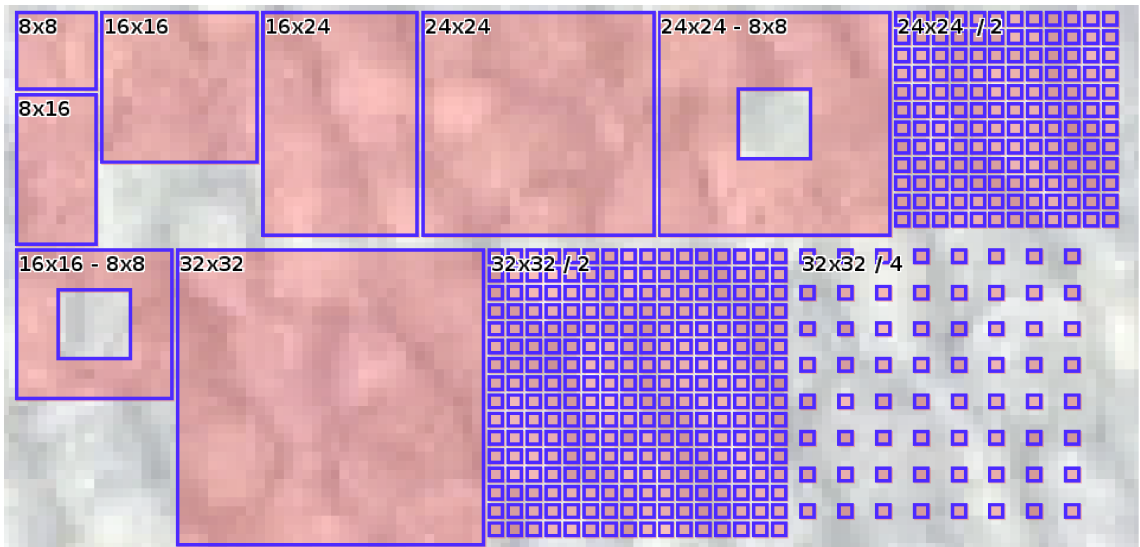


Figure 6.3: Shapes and sizes of the features that are investigated. The highlighted portions show the size and arrangement of the cells being considered.

Figure 6.3 shows the various shapes and sizes that were tested using the same benchmark attributes as the previous experiment to measure the effects on the algorithms using intensity and the difference in the intensity as the score. Note that the time consumption only covers the process of identifying the feature and not the tracking process, thus should be viewed relatively, as the correlation time will increase dramatically with an increased number of pixels. The number of candidates was increased to 256, as the small features were too fast to measure the running time accurately. Table 6.3 below summarises the findings.

The results show that by selecting between the maximum and minimum score, as discussed earlier, the uniqueness score could be maintained very high in most cases. The algorithms with obvious difference in the rank may have been due to the lack of variety in the sampling data or a particularly bad patch of ground texture. In any case, the modified feature shapes did not show significant signs of improvement, as the rate of finding the outlier using the proposed scoring algorithms were already quite high.

In both experiments, the utilisation score did not show any obvious relevance to the uniqueness score. It does, however, show the possible relationship between the effectiveness of the algorithm on different surfaces, as the uniqueness scores tended to be slightly lower when the utilisation score was lower. It may be feasible to run some simulations in the future to investigate any trends which may be present, which can be used to modify the feature size and shape.

As mentioned earlier, the processing time that was measured is only that of the feature selection process thus should be kept as small as possible. With this in mind, the 16 by 16 square shape was selected as the fixed shape as it also performed well in isolating the most distinct feature.

### 6.1.3 Shape and size

Table 6.3: Comparison of different feature shapes and sizes.

Algorithm	Carpet			Vinyl		
	Time (ms)	Utilization (%)	Uniqueness (%)	Time (ms)	Utilization (%)	Uniqueness (%)
Score	$\max(\sum  I_{x,y} - I_{ave(view)} )$					
8x8 square	2.98	5.27	98.07	5.78	0.72	92.23
16x16 square	5.63	1.85	99.65	4.05	1.57	98.97
24x24 square	18.06	0.88	99.84	17.39	1.03	95.73
32x32 square	32.95	0.99	99.79	30.07	0.98	99.43
16x8 rectangle	3.13	3.74	99.54	5.92	0.88	92.58
24x16 rectangle	7.86	1.65	98.78	5.62	1.12	99.18
16x16 – 8X8 donut	3.09	2.61	99.7	7.56	0.88	99.68
24x24 – 8X8 donut	17.03	1.22	99.65	16.68	0.57	85.59
24x24/2 spaced	3.32	2.19	62.5	3.96	1.76	92.76
32x32/2 spaced	6.97	1.12	99.72	6.37	0.97	99.3
32x32/4 spaced	2.81	2.77	99.86	3.12	0.48	97.94
Score	$\max(\sum  I_{x,y} - I_{x+1,y}  *  I_{x,y} - I_{x,y+1} )$					
16x16 square	22.41	1.01	99.41	19.49	0.018	99.71
24x16 rectangle	31.88	0.57	99.13	31.86	0.015	89.84
16x16 – 8X8 donut	17.69	1.31	99.77	18.3	0.5	72.58
32x32/2 spaced	20.96	0.85	99.67	20.66	0.029	99.1

With the ideal feature identified, a copy of the region is stored until the next frame is captured. The feature is then searched for within the next frame to determine the displacement of the feature. The opposite process of storing the entire frame, then finding the current feature in the previous frame can allow adjustment of the features until a good match is found, but requires a much larger footprint and processing time to be included for the backtracking algorithm.

## 6.2 Feature tracking

The core of the feature tracking algorithm involves a simple region alignment algorithm based on the distance measures between the attributes of the feature and the current view. Instead of investigating the different measures to consider the distance, such as the edge strengths and histograms (Huang et al., 1999; Pass et al., 1996), the intensity information was used due to its simplicity, adaptability in case the texture pattern is slightly modified between frames, near consistent locality, as well as the consistency with the feature selection process. Although the absolute values are susceptible to shifts from ambient light changes, this effect should be quite small due to the customised configuration using the sole light source.

Due to the small time interval between frames, the most frequently occurring changes to the feature is the interpolation effect between adjacent pixels when sub-

## 6.2 Feature tracking

pixel motions occur. This can invalidate the stored feature information, as the distinctive pattern may not appear within the view. The tracking algorithm must continue to operate even when the typical scores of the region alignment fluctuates and must attempt to differentiate between sub-pixel motion and the incorrect selection of the feature location.

Instead of simply scanning for the pattern in a linear sequence, the scan pattern can be controlled to increase the efficiency of the algorithm by finding the most potent candidates quickly. Since the majority of region alignment algorithms are based on a score accumulation approach, this can allow dynamic thresholds to be set for early termination instead of continuing to evaluate a badly matching candidate.

### 6.2.1 Motion prediction

Closed-loop techniques which make use of the feedback information to anticipate the current state, such as Kalman filters (Kalman, 1960; Welch & Bishop, 1995), allow effective modelling of the motions by predicting the likely state of the robot in the subsequent frames (Crowley, 1995; Ghahramani, 1998; Guivant, 2002; Negenborn, 2003; Roumeliotis & Bekey, 1997). Several attributes can be made use of to assist in the process, such as dead reckoning estimates from the motor command given to the wheels by knowing the dimensions of the wheels, the encoder feedback counts, the velocity from the wheel rotation and the clock, acceleration from the change in the velocity, and the location of where the feature was captured within the frame. Some of the measurements are maintained internally, while others require extraction from different modules of the robot, thus an appropriately synchronised or time-stamped measurement is required to make use of the information without corrupting the calculation.

Due to the noticeable latency in receiving the encoder values, the feedback information from the wheels was not used as part of the motion prediction algorithm. The information it can provide, however, relies on a precise motion model between the motor rotation and the motion of the robot. This involves a precise knowledge of the attributes involved in the motion, such as the wheel's circumference under compression, slippage, backlash and the follow through motion for non-breaking motors. This is especially problematic in many natural environments where these conditions can vary significantly. The additional information can thus be treated with a lower weighting to avoid potentially corrupting the prediction if they are required.

Since the process between sending the motor command and the motion eventually being carried out by the wheels is reliable, with the exception of overridden commands, the actual motor command that is generated by the high level processes can be passed onto the motion predictor to aid the algorithm. This information is especially useful when the search area is large due to a rapid change in the acceleration.

This implementation involves the use of a command queue which stores a small list of sent commands to be able to construct the anticipated motion of the robot. A list is required here due to the latency between sending the command and observing the motion. Since the latency can vary depending on the processing load, the commands are removed when there is a noticeable change in the command pattern,



### 6.2.1 Motion prediction

which is can include the detection of a change in the direction and when the size of the queue becomes a larger than a threshold value, which indicates that the there is too much delay or a command has been missed or misinterpreted. One way to manage the commands within the queue is to compress the commands of the same direction, which can be simplified by the restricted variety in the commands that can be sent. The current implementation caters for this by simply using the direction of the motion to add to the anticipated location.

There are three other attributes used to determine the location of the feature in the subsequent frame. They are the location where the feature was captured, current velocity and the acceleration vectors that are determined in the previous cycle. The velocity is determined by the change in the robot position from the last frame, while the acceleration is calculated from the difference in the velocity. The short term dependencies of these values mean the values can fluctuate quickly. This allows for quick adaptation to the rapidly changing velocities, as well as catering for when sudden motions occur, while still considering the historical information to encourage smooth motion. Note that these values are not time dependant, although the frame rate should remain reasonably consistent due to the process cycle and the capture rate, which can potentially introduce errors to the motion prediction. Figure 6.4 below illustrates the four components which contributes to the motion prediction.

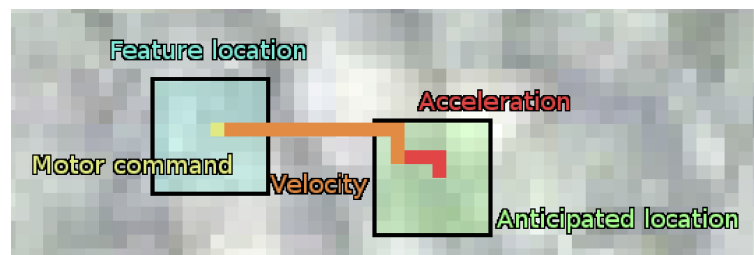


Figure 6.4: Illustration of the vector components predicting the robot motion. The different sources of motion prediction are applied individually to identify where the feature should be searched.

The errors in the prediction that are included in Kalman filters are indirectly corrected within the two vectors, since the displacement values are based on the precise observed motion. The prediction only has a very small and indirect influence on the accuracy and more to do with the time consumption, thus the effects of incorrect predictions are negligible. The occurrence of errors in the prediction is quite frequent since the robot frequently makes adjustments to its trajectory and also due to the discrete motion distances that can be observed. Although the actual velocity model is more of a sigmoid shape, the simple prediction model still allows for a good estimation of the robot's motion.

Quantifying the effects of the prediction can be done in several ways, such as the rate of a perfect match, average error distance in the prediction, the memory footprint, and the time taken to execute the new scan pattern. Although the most practical measure would be a time based measure to see if an improvement was made in the processing time, there is little meaning without a very high rate correct prediction. A more useful measure, the average error value, is determined to illustrate the overall correctness of the prediction. Figure 6.5 shows a sample histogram of the error rate of the algorithm when the robot is under operation for approximately 2

### 6.2.1 Motion prediction

minutes.

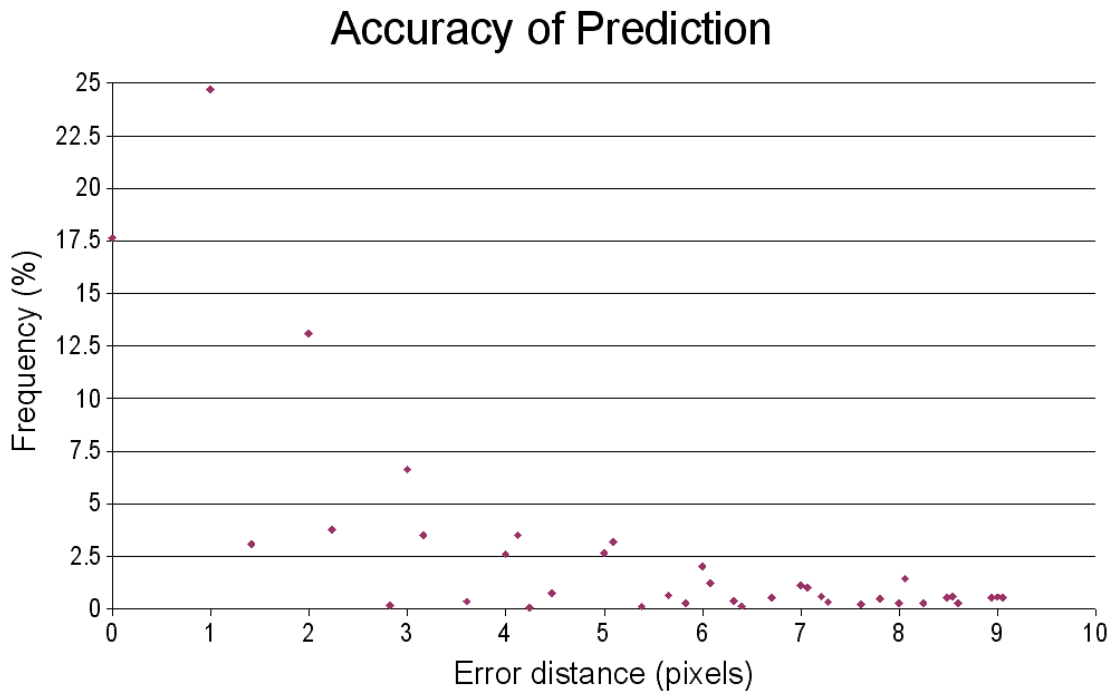


Figure 6.5: Histogram of prediction error rate.

The hit rate shows that the majority of the prediction errors are quite small.

As figure 6.5 indicates, the successful rate of prediction is quite high compared to those where the predictions have failed by a significant amount. An interesting observation is shown where the rate of a perfect prediction is lower than those where the prediction is off by one pixel width. This is partially the result of an incorrectly modelled motion, such as where the acceleration changes and some sub-pixel motions that are not accounted for. The other contributing factor, which will be described in more detail later, is the lack of activation of the algorithm when no motion is detected. Nevertheless, the algorithm shows promising results in anticipating the current feature location.

### 6.2.2 Radial scan

The resulting location can then be used as the starting location for the region alignment. Presuming that the prediction is reasonably accurate, the feature is likely to be found at or around the expected location. The traditional scan pattern involves a raster sequence, where small coordinate changes are applied to allow simpler and efficient data access and execution of code. Note that due to the unique alignment of each pixel involved when comparing the two groups of pixels, the difference accumulating approach used in the feature score evaluation cannot be applied here. Since the goal is to find the best fitting position, the scan pattern can be changed to prioritise the positions that are closer to the predicted location instead.

To arrange the sequence such that the positions are checked in the order of increasing error distance, a radial scan pattern is introduced. This shape must also

### 6.2.2 Radial scan

consider several other factors, such as likelihood of scalar and axial errors in the prediction, un-modelled environmental influences, and also the efficiency of generating and traversing the non-sequential coordinate offsets. Although the motion of the robot is used in the prediction, the proportional aspect of the motion in the two axes is not included in any adjustments. The domain knowledge about the motion patterns of the robot can be used to modify the scan pattern to anticipate more drifts in one axis over the other. Since the robot's primarily motion is in the longitudinal direction, the scan pattern can be changed to an elliptical shape, where the longer portions of the ellipse align with the direction of the motion. The shape reflects the tendency for more un-modelled contributors to the robot motion given the direction of the traversal. These include issues such as the rocking motion due to backlash, sudden acceleration from compression and decompression of the tyres from bumps and the change in surface friction, and slippage.

The sequence of the scan must be dynamically changed depending on the various state of the robot, which requires consideration to the additional costs involved. This generation at every cycle leads to a significant amount of processing load (Eberly, 1999), thus a lookup table is implemented to first generate the scan pattern offsets in a pre-processing stage before the robot starts its operation. Since the number of variation to the pattern can be quite large, this can lead to an extremely large memory footprint and potentially cause paging issues. However, since this motion prediction's primary purpose is to hasten the process of finding the best alignment of the feature, the algorithm is allowed to take short cuts to increase the efficiency over the accuracy in the optimal result. With this in mind, the shape of the scan pattern is fixed to the most common shape and the coordinate offsets of this are used for a simple lookup at runtime. Figure 6.6 illustrates the scan sequence by using the offset lookup table.

					139	131	123	115	104	106	116	128	136						
		135	114	103	89	81	73	67	58	64	68	74	86	100	107	132			
	122	99	85	63	53	43	37	29	24	26	34	40	48	60	82	92	117		
127	98	80	57	47	33	21	17	11	6	8	12	18	30	44	54	75	93	124	
113	91	72	52	39	23	16	5	3	1	2	4	13	22	38	49	69	90	108	
126	97	79	56	46	32	20	15	10	7	9	14	19	31	45	55	76	94	125	
	121	96	84	62	51	42	36	28	25	27	35	41	50	61	83	95	118		
		134	112	102	88	78	71	66	59	65	70	77	87	101	109	133			
					138	130	120	111	105	110	119	129	137						

Figure 6.6: Illustration of scan sequence using the radial scan. The numbers indicate the search order, which starts at the center and expands out in a clock-wise direction.

Due to the limited motion capability, the upper bound on the distance was used to restrict the scan pattern size. This allowed the offsets to be generated during the pre-processing stage by iterating through all longitudinal-latitudinal pairs and ranking the resulting distance. The elliptic shape meant that the symmetry can be exploited to only generate a quarter of the offsets. Since some offsets have the same distance measure, the ranking will be fixed in a predefined sequence when they are stored in the lookup table. Although it is not critical, as this is carried out in the pre-processing

## 6.2.2 Radial scan

stage, the distance measure does not have to be square rooted, as the magnitude is used to rank the offsets.

Due to the requirement that the pixels within the feature and the candidate must be correctly aligned, strategies such as partial matches cannot be used to modify the scan sequence. However, one possible case where this may be applicable is when there is a reasonably high level of correlation, but there is a slight mismatch in all of the pixels. This is likely to be caused by sub-pixel motion, thus finding a better candidate within the immediate neighbours may be possible. Rather than using irregular sequences, which can introduce extra levels of complexity and overheads, the scan sequence is left as consistent.

With the sequence now defined, the algorithm requires a mechanism to determine if the target feature has already been found. Since the indicator of this is the exact correlation in the intensity values, this is unlikely to occur. The algorithm instead determines when a better candidate cannot be found. This early termination approach is achieved by maintaining the best correlation score, which can be accumulated during the correlation, such that as soon as the score of the current candidate reaches past that point, it can be discarded and the next candidate in the sequence can be compared.

To measure the performance, the algorithm was compared to a simple linear scan pattern with and without using the threshold value. The test was carried out using a feature size of 16 by 16 pixels, where the search area was an ellipse with longitudinal range of  $\pm 10$  pixels and latitudinal range of  $\pm 4$  pixels on a carpet surface and was averaged out over a period of approximately 1 minute. To compare the efficiency of both algorithms equally, the same number of positions were compared as candidates. Since the radial scan traverses through 135 different positions, the linear scan was conducted over a 15 by 9 region. To make sure both algorithms performed consistently, the robot's speed was reduced to keep the feature displacement within 5 pixels. The results of the experiment is summarised in table 6.4 below.

Table 6.4: Processing time in linear and radial scan patterns with and without early termination.

	Without threshold (ms)	With threshold (ms)
Linear scan	7.866	6.964
Radial scan	10.166	5.624

As the results show, the overhead in the non-sequential offsets is quite large as the mirrored offsets were generated at runtime. However, by implementing the threshold algorithm, there was a dramatic improvement in the processing time, as the best candidates were found very early in the process.

While the effectiveness of the motion prediction plays a significant role in the performance improvement of the radial scan, it is also affected by various surface textures. In the presence of higher variation in the scores, which is related to the utilisation scores used in the earlier experiment, the scores are more likely to accumulate past the threshold value quicker, thus improving the effectiveness of the algorithm. As the typical ground texture consists of small amount of fluctuation, it is important to keep in mind of the relationship between the overhead and the number

### 6.2.2 Radial scan

of comparisons saved. A possible approach to target this in the future may be to make better use of the colour information, such as using a combination of the RGB and HSL scale (Cumani, 1989).

While using the threshold value has shown to improve the efficiency of the algorithm, the scan must continue to the end of the spiral to make sure no better solution can be found. An alternative to the radial scan is to apply an extra cost to the correlation score as the error in the prediction increases. This encourages the candidates that are far away to fail earlier as the ideal match is likely to have been found already. This also assists in disambiguating between similar features and encourages a smoother motion being observed. One very important drawback to this approach is that if the added cost is too high, it can suppress the ideal candidate and restrict the motions from being registered.

Initial results using an offset value showed that it either biased the correlations too much or none at all. Due to the rapidly fluctuating nature of the ground textures, selecting an appropriate offset becomes a difficult task. Rather than investigating the optimal weight, which is likely to be derived from the range of correlation scores that are being observed for the current surface, the biasing is removed in the current implementation and will be left for future investigation. If two candidates both end up having the same score, the one closer to the prediction is chosen.

### 6.2.3 Sub-pixel motion

The approaches involving the accumulation of the local coordinate and orientation changes are often criticised for their lack of error correction mechanism from small errors that are accumulated over long traversals. The use of external pose indicators, such as landmarks and measurements from other systems, can assist in re-calibrating the pose. However, the limited access and availability to such techniques, caused by visibility problems or when exploring new territories where there are no existing model to make use of, does not allow the mobile robot to depend on them at all times.

Often times, the errors are unnoticed by the algorithms, thus leading to warped poses where the robot's internal and actual pose gradually becomes significantly different. By estimating the errors, localisation algorithms can allow adjustments to the pose measurements, especially the confidence values, to flag the subsequent observations as potentially erroneous. It is also possible to make some corrections to the errors by taking other measurements with the available sensors to disambiguate the readings. Many approaches leave this task to the global localisation portion using landmarks or map correlation techniques to correct the pose.

In the presence of repetitive textures, blending of intensities from interpolation, as well as noisy artefacts, the feature tracking algorithm is constantly tested for reliability and accuracy in difficult circumstances. The proposed algorithm is able to perform quite effectively even when similar patterns appear by appropriately setting the feature size, frame rate, exposure time and the tracking algorithm. However, the algorithm is frequently troubled by the inability to find the exact position of the feature due to the discrete levels in the feature positions (Turkowski, 1990). Although the algorithm is often able to find the adjacent position to the actual feature position,

### 6.2.3 Sub-pixel motion

the small amount of error accumulates to a significant amount of error over time and must be reduced before it has a cascading effect.

One approach that is proposed is to store several copies of the feature which have been interpolated with its neighbours at various proportions to simulate the effects of sub-pixel motion. The region alignment algorithm can then match against the original feature and the interpolated versions to determine the best candidate. This naïve approach dramatically increases the search space for the feature, especially when considering multiple weights to generate the various interpolated versions. It also has the potential to introduce more errors from the introduction of implausible and fake features which may score well, but should not actually match.

Several improvements can be made to the above approach, starting with the reduction of the search space by controlling the activation of the sub-pixel motion checks. By specifying appropriate conditions for triggering the sub-pixel motion check, such as when the predicted location is ranked high, which can indicate a partial match, when the rank of the prediction is very low, which can be the result of large amounts of interpolation on highly repetitive textures, or when the best match is significantly far away from the anticipated location, the number of checks against the interpolated features can be reduced, as well as helping prevent false positives in the feature's location.

Implementing the ranking of the top few matches can be done quite simply by maintaining the top few matches in a sorted manner and using an insertion sort like algorithm of adding from the worst scored candidate's end, as the rate of encountering the best candidate is lower than encountering the almost best candidates. Since the algorithm only requires a small number of candidates to check for sub-pixel motion, the other candidates can be removed from the list. By using the threshold approach, the value to be compared against will be the worst candidate in the list, which will be discarded if a better candidate is found.

Another improvement can be made by storing the surrounding pixels of the feature instead of pre-generating the interpolated versions. This will allow the generation of interpolated feature with any weight when desired. An alternative is to interpolate the current view instead of the feature from the previous frame, as this will help reduce the memory footprint size. However, both these approaches will require knowledge about what weights to use or would face similar processing load issues as before.

Using the group of best matches, an alternative algorithm can be implemented by merging together the neighbouring matches. After the top few matches are determined, the distance between the best match and the others can be checked to see if they lie directly next to each other. If so, the matches can be merged together depending on the difference in the rank or the closeness of the correlation scores. The weighting used to interpolate them can also be controlled through the scores to determine the approximate positions of the feature.

Although these approach caters for a more flexible weighting of sub-pixel motion, it can potentially register sub-pixel motion where non existent. The erroneous measurements occur quite frequently, as the intensity characteristics are naturally interpolated across neighbouring pixels. This results in the neighbours having similar texture and thus also being ranked high. Instead of simply merging the highly ranked

### 6.2.3 Sub-pixel motion

candidates, the newly construct candidate and the correlation score can be compared to verify the correctness of the merge.

The use of the landmark can allow certain sub-pixel motions to be detected as it is tracked over multiple frames to narrow down the potential pose. However, due to the limited view area and the rapidly moving ground texture, the criteria for a long term landmark is very difficult to achieve. One of the conditions which can allow long term landmarks to be found is when the robot's motion is very small. When no motion is observed, the ground texture may have undergone a very small movement which was not registered. However, if the feature is re-captured, the very small motion is lost without any way of recovering from it. Instead, the feature captured from the previous frame can be re-used until the small motions accumulate until the motion becomes distinctive enough to be registered.

The simplest implementation in obtaining a higher precision in the motion tracking is to make use of a higher camera resolution to start off with. The additional pixels allows for a more detailed pattern to be captured, which can simulate the sub-pixel motion when the granularity is changed. The additional information does require more resources to process, while the transmission and memory copying process are often the bottle-neck when analysis is conducted on a reduced portion of the image. This can sometimes cause frames to be skipped and introduce delays in the other processes. As some of the USB cameras tend to apply software enhanced high resolution capturing and harsh compression algorithms, this can introduce more noise when observed at an individual pixel level and places more stress on the image restoration filters that are used.

The current implementation of the various sub-pixel motion strategies involves a combination of some of the approaches discussed above to provide a more robust algorithm in detecting the sub-pixel motion and attempting to register the actual motion.

When the feature is chosen, the pixel intensities of the feature and the immediate surrounding pixels are stored. The weights being used to interpolate them will vary, thus storing this will allow the tracking algorithm to later determine how the feature should be viewed. When scanning through the candidates for the feature motion, the top three candidates are maintained. The small number allows the thresholding to remain efficient while still allowing the candidates to be combined in both axes. An additional criterion is used in the merge, which restricts the second best candidate and third best candidate to only merge with the best candidate and must merge in the perpendicular direction to each other. This eliminates the case of implausible sub-pixel motions.

To utilise the confidence of the candidate, the correlation scores are used to weight the interpolation. This is done by dividing the score of the best candidate with the sum of the best and the other candidate. The value is then applied to the coordinate point of the best candidate, as well as the weight to derive the new intensity values, which is then correlated and compared against the best candidate.

To check to see if the actual motion was near the predicted location, the predicted location and the four adjacent positions are interpolated using a predefined weight of 0.5 to generate four more candidates that are also included. This acts as a fall back mechanism in case the ideal location was skipped due to badly interpolated

### 6.2.3 Sub-pixel motion

intensities. Due to symmetric weights, it is possible to efficiently generate the interpolated values by re-using the merged values for the extra candidates. Note that this algorithm can still be executed even if the best candidate was found where it was predicted to be.

With the components of the tracking algorithm defined, the overall accuracy of the visual odometry can be compared. Table 6.5 shows the results of moving the robot for 5m in a straight line on a carpet floor. To measure the difference in the precision, the experiment was carried out for two camera resolutions to determine the reliability using a higher resolution. Note that to maintain the consistency between the algorithms, the operational speed of the robot was modified. This meant the difference in the processing load was isolated to the handling of the image data stream and not the algorithm itself.

Table 6.5: Precision of the visual odometry using different sub-pixel motion correction techniques.

<b>Algorithm</b>	<b>Error (%)</b>	
	<b>Resolution</b>	
	<b>160 x 120</b>	<b>320 x 240</b>
None	5.5	1.93
Merge top candidates	1.43	1.41
Fixed weight with neighbour	5.2	1.67
Variable weight with neighbour	1.02	0.32
Combined	0.98	0.28

## 6.3 Motion detection

With the motion of the feature determined, the vector can be converted to determine the motion of the robot. The triangulation approach allows this instantaneous motion to be derived and accumulated, but is limited to only translations due to the lack of rotation.

Since the algorithm uses an accumulative approach, the robot must ground the coordinate point to some known location. At this stage, no other sensors or algorithms have been introduced to take advantage of well recognisable landmarks, thus the starting position of the robot is used as the origin.

The algorithm currently does not make use of a long term feature to calibrate its pose, thus the accumulative approach is still prone to drifting in the tracking and camera configuration errors that can quickly accumulate. To help reduce the configuration errors, it is important to make frequent calibration measurements to improve the accuracy of the configuration attributes so they converge to the actual value.



### 6.4 Summary

The proposed feature tracking algorithm considers many practical aspects of ground texture tracking for visual odometry. In doing so, it incorporates many constraints that are based on the environment and the task. This allows efficient utilisation of the available data and resources to improve the accuracy and efficiency of the algorithm for real time application. The notable constraints placed are that the view's proportionality remains the same, the ground texture contains distinguishable patterns, and that the operational speed of the robot remains controlled to allow the captured texture to show up in the subsequent frame.

The various techniques provide a reliable and efficient approach in tracking ground textures, which forms the basis for the visual odometry algorithm. The components that were considered include the selection criteria for the feature, the number and arrangement of the individual pixels that make up the feature, motion estimation using feedback information to improve the efficiency of the search, modified scan patterns to encourage early detection of the ideal candidate, and also several algorithms to reduce and recover from sub-pixel motions due to the discrete amount of change the camera can determine.

Although the majority of the testing of the above approaches were conducted on a set of known surfaces, preliminary tests showed that the algorithm can operate quite well on many other surfaces that are typically seen in indoor environments. A more comprehensible experiment on this is carried out in chapter 7, which do not require the modification of the attributes determined here.

During the testing phases, the tracking showed infrequent occurrences of large jumps away from the predicted area. This behaviour was commonly seen when frames were skipped or when motion blur had occurred due to bumps or jerks of the robot, especially when using the higher resolution. This meant extra precautions had to be taken when setting the camera configuration attributes, such as the frame rate and exposure time to reduce the occurrence of this problem.

The approach investigated the effectiveness of a single, non-rotational feature tracker of ground textures, which has showed promising results in observing the pose changes of the robot. In its current state, the technique has too many limitations to be used as the sole technique for the mobile robot localisation. To target this, the technique must be integrated with other approaches and modules to improve its effectiveness and accuracy. The strategies for this are discussed in chapter 7.

## Chapter 7 – Multi-sensor localisation

There are many different approaches to tracking the pose of a mobile robot. The approaches can be separated into two broad categories, where one use of an observer to monitor the motion of the robot, and the other being the robot measuring the motion of the environment with respect to itself. The two approaches can differ in precision, reliability and also on the availability, since the observer must be placed within the environment before the mobile robot and know its own pose.

To track the pose from the robot's perspective, the location and orientation can be derived by monitoring how the robot has moved through dead reckoning or aligning itself against landmarks. The landmarks can be dynamically determined or given to the robot as markers to look out for, but in doing so, requires interaction with the environment before the robot, thus limits the capabilities in similar ways to having the observer. By selecting the features dynamically, it places an extra burden on the localisation algorithm to identify and re-locate the landmark from different perspectives. This can often only allow for local pose information to be derived and require multiple landmarks to be viewed simultaneously to accurately triangulate its position. The simultaneous tracking and fusing of multiple landmark tracking introduces many new issues on top of the standard tracking problem.

### 7.1 Multiple tracker

The use of the tracker introduced in chapter 6 has shown promising capabilities for mobile robot localisation, yet the constraints which improve the efficiency restrict the detectable motion to simple 2D displacements. For it to be a useful localisation technique, the rotational motions must also be captured to allow the robot and world coordinate axes to be correctly aligned. By introducing a second feature tracker on the robot, the two detected motions, as well as the constraint between the positions of the two trackers can be used to determine rotational changes. The two motion vectors that are detected can be used to determine the instantaneous center of curvature (ICC) of the motion, while more than two can introduce redundancy to improve the accuracy or disambiguate the inconsistencies in the detected motions.

Although the introduction of more trackers introduces many new capabilities, extra problems are introduced, such as the ideal location of the trackers, synchronization, as well as the extra processing load. Many of these issues are discussed and strategies are introduced to construct a localisation technique capable of detecting translation and rotation.

#### 7.1.1 Testing and validation

Throughout the development process, the various algorithms were tested using a consistent set up, which comprised of a translational test of traversing forward and backwards along a guided rail, and a rotational test around the center of the robot

### 7.1.1 Testing and validation

using a lazy susan. The purpose of the guides is to maintain a consistent motion for the different implementations to be compared. Although the motions are not entirely realistic, it allows the different algorithms to compare their performance evenly as if under ideal motion conditions. A more realistic test is carried out later on using the derived localisation algorithm.

The rail was placed near the center of the robot for it to slide against, but due to the imperfect set up, it allowed for slight rotations to occur. The rotational test was also not perfect, as the robot had to be lifted slightly for the rotating plate, thus the amount of ground contact was not normal and the camera height parameter had to be adjusted. With both tests, the placement of the guides was done by manually observing the robot's motion, which may have contributed to some errors in terms of the expected motions.

The tests were conducted on a flat carpet surface using a tape measure and a protractor with 0.5 mm and 0.5 degree of precision respectively. The forward and backward test was conducted by moving forward for 1 meter, then reversing along the same path. The rotational test was done by measuring the algorithm's performance over a 360 degree rotation, then reversing the same amount to get back at the original orientation. The accumulated pose was then compared to the expected motions to determine the accuracy. This was carried out for five repetitions each, then averaged to indicate the consistency of the performance.

The purpose of reversing the same path allowed the removal of any scale based inaccuracies which many not have been ironed out. It also has the effect of traversing over the same ground pattern, thus potentially reversing or doubling any effects of error prone areas. Figure 7.1 illustrates the testing set-up.

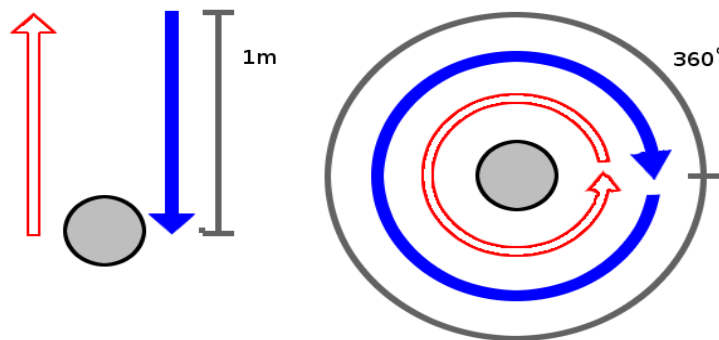


Figure 7.1: Testing configuration.

The circles represent the robot, the outlined arrows indicate the forward and clockwise motion tests, while the solid arrows represent the reverse and anti-clockwise motions.

### 7.1.2 Single camera

Techniques involving the tracking of multiple features are commonly applied in machine vision, where the extra trackers allow the algorithm to focus on several different entities simultaneously, or to provide a better approximation of the motion by a voting or interpolation algorithms, such as approaches which use optical flow (Bretzner & Lindeberg, 1996; Camus & Bulthoff, 1995; Irani et al., 1997; McCarthy,

### 7.1.2 Single camera

2005). Depending on the criteria of the tasks, the approaches and techniques used can vary significantly to optimise the resource usage and improving the quality of the measurements.

Many of the issues with regards to identifying and tracking ground textures has already been considered, thus this portion of the algorithm can simply make use of multiple instances of the developed tracker. Since the processing requirements for these trackers are slightly more expensive compared to simple trackers., this can limit the number of instances being used simultaneously, but at the same time, has a much higher reliability. The right balance between the two should ideally be used, but several criteria must also be considered during the design.

When attempting to support the measurements of the other tracker, it is important for the tracker to be exposed to a very similar type of motion. This means that the location viewed by the tracker must be positioned close to each other, as well as being away from the center of curvature. Since a small shifting of the viewing window for the feature is all that is necessary to produce a new feature, it is possible to capture the second feature that is sub-millimeters away from the first feature. This allows the two trackers to capture the motions with minimal difference in the motion vectors, which can be easily combined without the hassle of location dependant pose changes. However, having closely placed trackers cause the majority of the texture to be overlapped with each other, thus the redundant measurement is also likely to be corrupted if the error was due to a bad region of indistinguishable ground texture. To increase the reliability, the tracker should thus be located slightly away from the other tracker to avoid observing the same bad texture. This also increases the chances of either one encountering the bad texture, but the approach allows the option of verifying the measurements when discrepancy occurs.

The commonly used approach for verification is to implement a simple voting algorithm using odd number of measurements. Since this approach requires many redundant resources to implement, another common approach that is used is to make use of a score for each measurement to indicate the validity or the confidence. This type of redundancy mechanism is suited for trackers that are not so reliable by themselves and require multiple, concurrent trackers to support the measurements. However, by using a reliable tracker, it reduces the need to rely on supporting measurements from the similar ground texture motion.

By changing the distance between the trackers, the dilution of precision can be controlled to improve the precision of the derived motion. Placing the trackers at extreme ends of the viewable area of the camera will maximise the distance between the two trackers, but considerations must be made for the quality of the images observed at different points within the image and the typical motions encountered by the trackers. Note that increasing the mounting height of the camera does not improve the relative accuracies as the actual distance between the trackers remains the same.

As noted in chapter 5, although small, the quality of the image degrades as they near the outer edge of the viewing area due to warping. Since the distance between the trackers is quite small in comparison to the typical distance to the center of curvature, even a slight mismatch in the scales can result in a large deviation of the motion measurement. The cropping approach introduced earlier reduces the effect of

### 7.1.2 Single camera

the warping, but limits the available positions for the trackers to be used.

Another critical issue to consider is the importance of synchronisation between the multiple trackers. When using a single camera, it is quite simple to obtain a frame where all of the intensity measurements were taken at the same time. This can be done by controlling the exposure time and removing any inter-frame interpolations, which can delay the transmission and cause portions of the image to be out of sync with other parts of the image. This sampling problem is typically taken care of by the camera and the driver to produce a well synchronised image. However, the same capturing time does not mean the exact amount of motion will be registered due to the discrete amount of motion that can be measured.

Since the combined motion of the trackers is used to determine the overall motion of the robot, the difference can contribute to large drifts between the actual and measured motion. This problem is targeted by the sub-pixel motion tracker introduced in chapter 6, but the previous analysis did not consider the timing difference between the detected motions of the trackers.

On a related note, the timing difference experienced between the sending of the motor commands, the motors carrying out the motion, and the registration of the motion with the sensors, does not affect the precision of the current localisation algorithm. Since the delay only causes a mismatch to the information from other modules, such as the path planning or mapping modules, where the information must be time stamped for appropriate integration with other sensor data, this issue can be ignored within the localisation module and dealt with at a later stage. If an implementation of the localisation algorithm does require timing based information, such as a more precise velocity or acceleration model, this latency must be considered using techniques such as clock synchronization with time stamps on every event.

Due to the motion constraints enforced by the wheel configuration, each portion of the robot will experience different types of ground texture motion. This information can be used to select an appropriate location of the camera. As noted earlier, the precision can be made higher by increasing the distance away from the center of curvature. Since the center of curvature lies on a line connecting the two wheels for a differential drive system, placing the camera at the outer perimeters of the robot yields the most attractive tracker.

By placing the tracker onto the sides of the robot, they will experience a longitudinally dominant motion, even when the robot rotates. This means that the subtle latitudinal motions must be accurately registered to differentiate between rotation and displacement. Trackers placed at the front or the back of the robot will register distinctively different motions during rotation and displacement, thus simplifying the process when distinguishing between the two. Another possible location is somewhere between the longitudinal and latitudinal extremes of the robot. This will allow a mixture of motions, but since the majority of the motion will cause a combination of motion in the two axes; this can create more ambiguity due to a more jagged motion and interpolations in both axes. Figure 7.2 shows the various mounting location of the camera, as well as the motion vectors that are likely to be encountered for the corresponding wheel motion.

### 7.1.2 Single camera

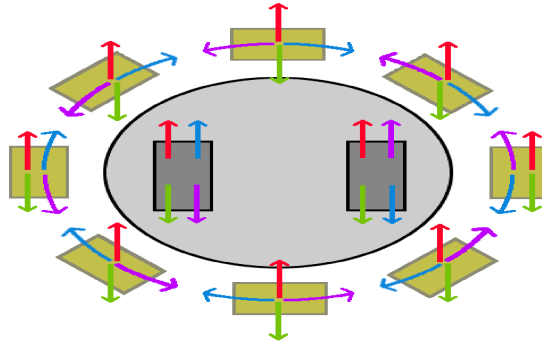


Figure 7.2: Camera mounting position and the motion vectors when rotation and translation occur.

The camera locations are shown as the olive rectangles, while the wheel motion and the corresponding observed motions are shown as colour coded arrows.

When deciding on where to place a component onto a robot, many practical considerations must be made. For example, although placing the camera at the front of the robot can allow the robot to foresee hazardous ground up ahead, it can potentially cause damage to the device if it accidentally collides with obstacles. Depending on the design of the robot chassis, it may be possible to place the camera within the body, thus protecting it from potential collision related hazards. The chassis of the mobile robot used does allow for a camera to be mounted inside, where it can look through a small gap towards the back of the robot, but required the camera to be mounted quite close to the ground to avoid portions of the chassis obstructing the view. This meant that the speed of the robot was severely limited and also required a camera with a small minimum focal distance. Placing the camera at the back of the robot provides the same range of motion as the front of the robot while greatly reducing the possibility of collision, as the majority of the motions are in the forward direction. Figure 7.3 below shows photographs of the cameras mounted on the robot. The internally placed camera is provided with a natural shielding, while the side camera mount provides a wider viewing area by stabilising the camera with adhesives.



Figure 7.3: Webcams placed within and on the side of the robot. Left image shows the eyeball camera just behind the battery, while the right image shows the configuration of the side mounted camera.

### 7.1.2 Single camera

One of the downside that was discovered in placing the camera at the back of the robot was that the search area for the feature had to be increased. Although this may seem like a small increase in the overall scheme of things, it also increases the likelihood of detecting a false feature due to the repetitive patterns.

The benefits of being able to distinctively determine the difference between a translation and a rotation is quite significant, as these two types of motions form the majority of the robot's traversal, the other being elevation, which is disallowed in the environment. Since the camera mounted on the side is only able to distinguish the difference using a small variation in the sideways direction, the approach falsely identifies a significant amount of rotation as translation motion and vice versa. As the results in table 7.1 show, neither of the approaches resulted in anything feasible due to the lack of distinctive behaviour between the trackers within the single camera. The accuracy of the camera being placed at the back of the robot was exceptionally bad due to the slight latitudinal motions that were observed. The experiment was carried out with the two trackers placed at the extreme ends of the image after the cropping.

Table 7.1: Accuracy of single camera tracking.

Position	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
Side	2.05	1.9	69.2	81.53
Back	53.16	55.53	62.43	0.89

### 7.1.3 Multiple cameras

An alternate approach to the above is to consider the use of multiple cameras to increase the possible arrangement of the trackers. This can increase the precision in deriving the center of curvature, as the motions that are detected can be made more distinctive. The extra camera must consider the issues described earlier, as well as several others including simultaneous access to the two devices which can possibly look identical to the media library accessing the camera, characteristic differences between the camera, the extra processing load, synchronization of the frames being captured, and the secondary location to place the camera.

Depending on the library being used to access the devices, the simultaneous use of webcams can be prohibited, such as with the Java Media Frame library and Video for Windows. Therefore, it is important to develop the application using an appropriate language and library to make use of the extra sensor. The current implementation has been ported over to C++, and makes use of the DirectShow library which allows simultaneous access to multiple devices of the same type.

Although the two cameras may be identical models, the subtle characteristic differences, including the lens quality, the responsiveness of the photo-sensors, and the noise ratio, can cause variations to the captured image. When the captured data is mapped to the same representation, it is important to apply filters, usually at the software level, to reduce these inconsistencies. This requires calibration processes to identify the camera characteristics, such as those introduced in chapter 5, and inter-

### 7.1.3 Multiple cameras

camera consistency measures by observing the same object features to note any differences in the captured frames. Due to the aliasing effects, this process must be carried out by taking multiple snapshots of the same feature from slightly different positions and comparing all pairs to find the best positional correlation. Slight variation in the adjustable characteristics like the mount height, exposure time and colour adjustments can all lead to similar problems, thus should be properly noted.

Issues with the processing load are one of the key deciding factor in making use of extra sensors or devices. If the sensors were allocated dedicated processors for handling the non-shared data, it is possible to utilise the extra information quite efficiently. However, since the processing of the webcam images are done on the shared processor on board the laptop computer, this causes a significant increase in the load and potentially slows down the other modules of the system. The change in the processing time can be seen in table 7.2. Note that these values should be observed proportionally, as the conditions between this experiment and those in earlier chapters are not exactly the same. Choosing the appropriate settings and algorithms becomes even more crucial and requires some sacrifices to be made, such as the reliability or the reduction of the search area for the tracker by limiting the operational speed of the robot.

Table 7.2: Time consumption using one and two cameras.

<b>Task</b>	<b>One (ms)</b>	<b>Two (ms)</b>
Initialisation	5.03	9.79
Frame capture	0.17	0.32
Execution	0.12	0.13

A related issue to the registration timing of the motions is the precise synchronization of the two frames that are used before they are combined as one motion. Since the capturing occurs independently at the actual devices, the two images can capture the scene at slightly different times. The rate of the capturing typically remains fixed, but can be shifted depending on the devices and processing load at the driver level. A simple approach at determining the capture time difference can be implemented by capturing the frames when a sudden change is applied to both the cameras and monitoring the difference. One way to implement this is to use a bright light source, where switching the light on and off in a dark room will cause instantaneous change to both observers. When doing so, it is important to set the exposure time to low and the frame rate as high as possible and to note any delays caused by extra filters that are active on the cameras. For example, when using a colour balancing filter, the images can be delayed as they buffer the adjacent frames to shift the captured intensity values.

Inconsistent intervals in the capture time is a much more difficult attribute to identify, as it requires regular calibration phases to detect and adjust the access times of the frames. This can sometimes be achieved by using a precise and synchronized clock together with interpolation. This can be provided by external light sources, such as the flickering of the ambient light, which can be used to calibrate the capturing intervals. However, dependency on these external calibration regulators simply passes on the responsibility of precisely keeping the time to another system



### 7.1.3 Multiple cameras

and also introduces some overheads in making use of the other system.

After numerous test runs to determine the timing difference between capturing of the images, it was noted that the capture times of both cameras were indistinguishable. Although the test was not comprehensive to say it never occurs, early results showed that the focus should be placed on other portions of the algorithm and avoid the overhead of detecting unlikely events.

When deciding on where to mount the cameras, it is vital to take advantage of being able to increase the distance between the trackers. Since the risk of placing the camera at the front of the robot remains the same and the problem of jagged motion still exists in placing the cameras that are not aligned with the longitudinal or latitudinal axes, these are once again ignored. By placing the cameras at the back and the side of the robot, the back camera is still able to distinguish the different types of motion, but allows the camera mounted on the side to support the measurement. It is also possible to allocate each camera a designated axes to focus on, such that the back camera is used to identify rotation first, followed by translation by the side camera if a rotation was not detected.

The initial problem with placing the camera on the side was that it was unable to distinguish the difference between translational and rotational motion. However, using the second camera allows simple disambiguation of the two types of motion by placing the camera on the either side of the robot. The combination of the two motions can be thought of as the equivalent to closed-loop version of using the wheel motions. When the two trackers support each other's motion, the robot motion can be said to be a translational motion, while if they conflict with each other, the robot has undergone a rotation.

As the motions compliment each other, it is important for each tracker to correctly and precisely measure the motion. This means the approach may require additional strategies to improve its reliability, possibly by utilising multiple trackers within each camera to allow error corrections to occur before the motion is combined with the other camera.

The two camera positions discussed above still showed unattractive results, especially in the rotational tests. The configuration with the camera set up at the back of the robot showed occasional signs of erroneous tracking, possibly due to the increased viewing area as the shape of the radial scan pattern had to be enlarged. Using the approach of dedicated motion checking for each of the cameras, the latitudinal motions that were detected by the back camera triggered a rotational motion to be detected, but also did not display a desirable algorithm. This approach also severely limited the types of motions possible by the robot as it assumes that the center of curvature will always be exactly in the center of the robot. The results of the experiment, which was conducted in the same way as the single camera approaches, are summarised in table 7.3.

The use of multiple cameras was not able to display attractive results, but showed slight improvements in the consistency. During the experiments, it was noted that the majority of errors were the result of mismatches in the motions that were detected by the two trackers. This was especially problematic when rotation was being observed, as the center of curvature that were being derived often did not reflect the true motion of the robot.

### 7.1.3 Multiple cameras

Table 7.3: Accuracy of multiple camera tracking.

Position	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
Back-side	22.55	6.61	41.99	3.42
Back-side dedicated	36.84	8.34	32.68	5.28
Side-side	2.88	1.21	82.12	9.2

## 7.2 Smooth motion

Not being able to precisely register the motion at each tracker is a significantly issue which must be addressed if the multiple trackers are to be merged together to derive the robot motion. It is possible to eliminate this problem under certain conditions, such as when the robot always moves a discrete distance or the camera can be moved around precisely while the robot remains stationary. However, when these camera motions cannot be controlled, the sub-pixel motions must be calculated or guessed by using the available information about the type of motion the robot is under.

### 7.2.1 Accumulation

As noted in the single tracker explanation, the accumulation of the sub-pixel motions will eventually result in a full pixel motion. It was also noted that delays between the robot motion and the registration of the robot were not a significant issue within this module. Based on the two assumptions, it is possible to accumulate the robot motions until a certain condition is met, which is when the motions are processed as a one large pose change. The motion can be applied in a single phase, or split into the average motions then applied retrospectively. Although this allows the majority of the sub-pixel motions to accumulate, there will still be some motion that does not accumulate enough during the build up stage. The proportionality of this, however, should be much less than those that do get accumulated depending on the condition that is used to trigger the batch processing. Before the appropriate triggering condition can be determined, the characteristics of the algorithm and how well it fits to the environment must be considered. Figure 7.4 below shows the motion vector accumulation algorithm, along with the pose differences it introduces.

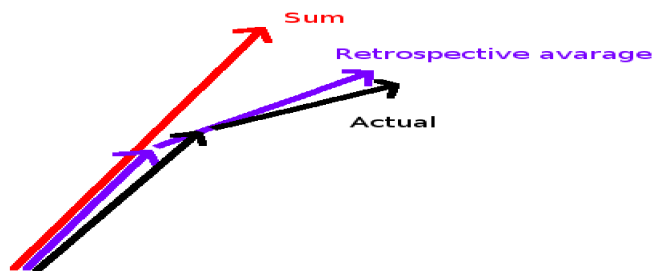


Figure 7.4: Motion vector accumulation approaches. The effect of merging multiple motion vectors before being applied can be observed by the difference in the final pose.

### 7.2.1 Accumulation

One of the key issues of the above approach is that the smaller motions that are accumulated are compressed and thus loses their individuality as well as any sequence information. The sequence in which the motions are detected can play a vital role, as the coordinate axes used can be misaligned with the actual pose of the robot. The extreme case of this is where the motion vectors cancel each other out and misses out on the crucial rotational changes. One way to reduce this is to specify a triggering condition so that any large change to the motion direction to the one currently being accumulated will trigger the accumulated motion to be applied and reset before the current motion is used.

The simplest implementation for motion vector accumulation involves the triggering condition being a counter, where a fixed number of measurements are buffered before they are combined and emptied. This will allow the localisation algorithm to stay reasonably up to date. However, as noted earlier, the approach also requires an additional triggering condition to avoid the introduction of misalignment errors. By observing the current trend in the motion, such as using the previous motion or the average of the motion currently stored in the buffer, the amount of error that will be introduced can be estimated to determine if the accumulation should occur or not.

To remove this error further, the exact orientation of the motion vector must be compared as well as reducing the motions capturing and application intervals, such as by making sure that all motion vectors are consistent. Since this constraint can be difficult to achieve, the accumulation approach must allow for certain amount of error being introduced. One reasonable approach consists of allowing the accumulation if the motion vector is in the same quadrant. This will avoid the case of the motions cancelling each other out.

Since all of the trackers must be synchronised, the approach must enable access to the other tracker's state of the accumulated motion vectors, as the emptying must occur at the same time. It is possible to postpone the updates of the pose until all trackers have applied some of the motions and modify the pose retrospectively, but this causes issues when integrating with other modules if certain guarantees cannot be made with regards to how long the other modules must wait.

The accumulation algorithm was first tested using various number of frames as the threshold value to trigger the emptying of the buffered motion vectors. The testing of this involved a slightly different process to before, where the robot was moved without the use of the physical guides. This allowed the characteristics of the approach to be measured in a realistic context to test the problematic areas of the algorithm. Although the approach provided reasonable results when the guides were used, the algorithm failed to register the subtle irregularities in the motions and was unable to perform adequately under normal mode of operation, which is summarised in table 7.4.

Instead of using just the simple counter as the threshold for emptying the buffer, the change in the motion vector orientation was also introduced to trigger the buffer being emptied. Defining the valid range of angles to allow accumulation can be difficult, as the changes in the orientation lead to differences to the actual motion. To simplify the condition, the accumulation was allowed if the direction did not cross the borders of the quadrant. To allow for a slower rotation, pure forward or backward

### 7.2.1 Accumulation

transitions were flagged as neutral and did not trigger the emptying of the buffer. Using just the above condition to empty the buffer could potentially result with extremely large number of motions being condensed into one if the robot continues to move in one direction. To maintain the regular emptying of the buffer, a count based threshold was used in combination. The results of the three approaches and the algorithm for distributing the accumulated motions with a buffer size of 8 can be seen in table 7.5.

Table 7.4: Effect of accumulating the motion vectors.

Buffer size	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
4	3.21	0.58	11.43	1.22
8	4.1	0.59	10.66	0.87
16	3.57	0.3	13	2.76
32	3.83	0.29	10.9	5.2

Table 7.5: Various trigger conditions and distribution algorithm for the accumulated motion.

Trigger condition	Distribution	Translation (%)		Rotation (%)	
		Forward	Backward	Clockwise	Anti-clockwise
Quadrant only	One	3.71	0.95	12.51	0.72
Quadrant + neutral	One	4.22	0.68	12.6	0.81
Quadrant + neutral + buffer	One	3.5	0.44	11.38	0.65
	Average	5.67	0.91	9.28	0.58

Note that in both experiments, the large improvement in the precision, especially those of rotation, is the result of applying a differential motion model, which will be explained in detail later on. The change was necessary to note the behaviour of this approach with a more precision error tolerant model, as the dependency on the motion vector direction was often too great for the exact motion models.

### 7.2.2 Window

An alternate approach to the batch processing of the motion is to make use of a sliding window to interpolate the registered motions and apply a portion of the motion at each cycle. Unlike the earlier approach, this allows for a much quicker response time for the registering of the motion while still allowing it to influence to subsequent motions as a way of simulating a smoother motion. Using a uniform weight in the window leads to a very similar effect to the buffering approach where the average of the motion being used. By modifying this weight, such that the more recent motions are weighted greater than older motions, the immediacy of the motion registration can be modified to improve the response time. Figure 7.5 illustrates the windowed approach, along with the weights for the entries in the window.

## 7.2.2 Window

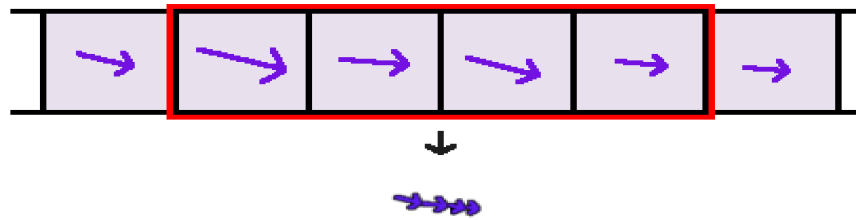


Figure 7.5: Windowed interpolation of motion vectors.

The motion vectors from consecutive frames are weighted down and combined to smooth the motions. Each square represents an image frame, the rectangle represents the window, and the arrows represent the motion vectors that were observed.

Similarly to the previous approach, the use of the window also suffers from delays in the registration of the motion. The weighting is able to reduce this effect, but is unable to completely remove this without a reset like mechanism, which can correctly approximate the current accumulation of the sub-pixel motions. A simple example of this problem is when motion can be observed even after the robot has stopped. Although the robot has moved that amount, the delay can appear unnatural with incorrect time stamps for the pose and indicate motions that did not actually occur.

As illustrated in figure 7.4, the more merging that occurs between the motion vectors, the greater the difference in the rotational pose it registers. The misalignment of the starting positions of the sub-motions being combined leads to an accumulation of pose errors, which is dependant on the proportional split of the motion vector. This is illustrated in figure 7.6 where the difference in the positions can be seen when dividing the motion vector into two segments.  $M$  represents the proportional weight used to split the motion,  $x$  and  $y$  represent the horizontal and vertical motion respectively, and  $\theta$  represents the orientation of the motion. The graphs illustrate that the errors are reduced when the proportion is increased for the first division.

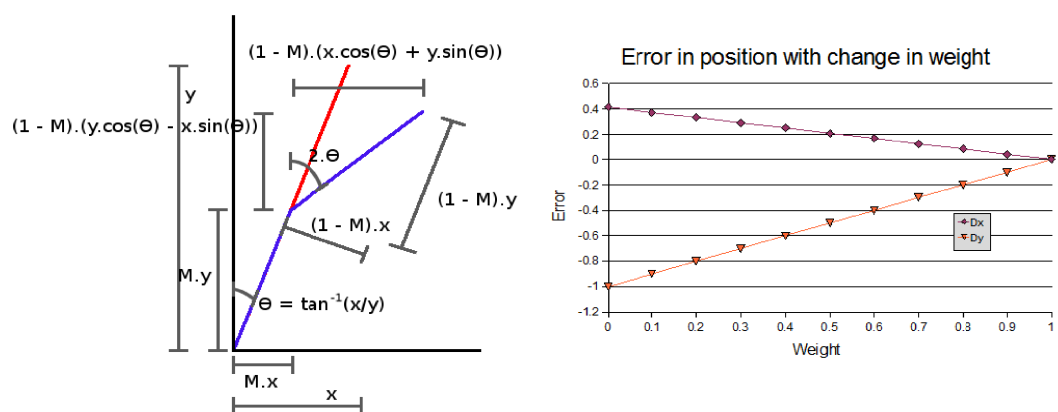


Figure 7.6: Error in segmenting a motion vector.

Left image shows the components used in the derivation of the pose, while the right image shows a sample relationship between the horizontal and vertical error with respect to the proportional weight. Note that the error for weight being 0 would actually be 0, as no rotation would occur initially.

### 7.2.2 Window

By steeply decreasing the proportion being distributed across the multiple frames, the accumulated error can be reduced. Using this approach assumes certain amount of error will be introduced, as the merging of the motion vectors are required to cater for the sub-pixel motion. The weights can be generated dynamically, or derived in the pre-processing stage and stored in a lookup table. Figure 7.7 shows four weight distribution functions that were considered for a window size of 4 where  $W$  is the weight,  $i$  is the index,  $N$  is the window size, and  $F$  is the weight factor. The derivation of the quadratic equation is shown below as an example.

$$\sum_{j=0}^{N-1} W_j = 1 \quad (26)$$

$$W_j = F \cdot W_{j+1} \quad (27)$$

Equation 26 can be expanded to:

$$W_1 + W_2 + W_3 + \dots + W_N = 1 \quad (28)$$

$$W_1 + W_1/F + W_1/F^2 + \dots + W_1/F^{N-1} = 1 \quad (29)$$

$$W_1 = F^{N-1} / \sum_{j=0}^{N-1} F^j \quad (30)$$

Substituting equation 30 with equation 27 yields:

$$W_i = F^{N-i} / \sum_{j=0}^{N-1} F^j \quad (31)$$

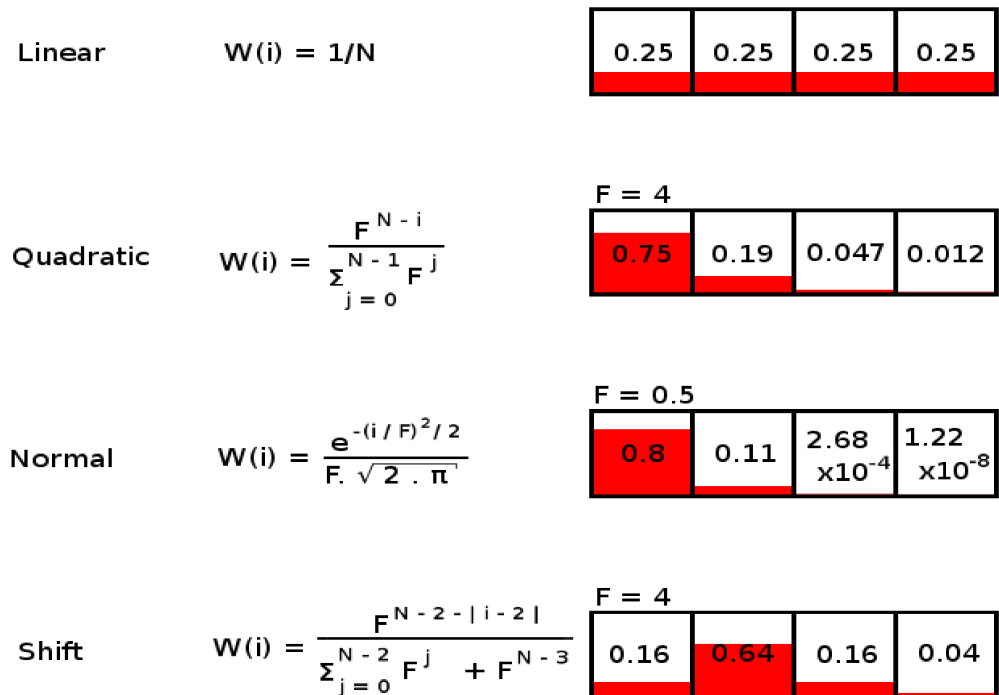


Figure 7.7: Weight distribution functions for a window size of 4. The left column is the name, middle column is the formula used to derive the weight, while the right column visually illustrates the weight distribution if  $N$  is 4.

## 7.2.2 Window

Although the windowed approach involves a slight increase to the processing load to calculate the weighted motion at each frame, the benefits of using this approach include the faster registration of motions and the improvements in the accuracy due to a more consistent motion being modelled. The summary of the results for the four weight functions and several window and weight factor sizes can be seen in table 7.6.

Table 7.6: Windowed motion with various weight distributions and sizes.

Distribution	Factor	Size	Translation (%)		Rotation (%)	
			Forward	Backward	Clockwise	Anti-clockwise
Constant	1	4	3.1	0.86	10.87	1.37
		8	3.98	0.88	11.65	1.08
Quadratic	2	4	5.23	1.03	12.2	2.01
		8	4.8	0.89	11.9	1.34
		16	4.91	0.91	12.42	1.27
	4	4	2.88	0.78	9.72	0.97
		8	2.91	0.87	9.84	0.89
		16	3.01	0.8	11.21	1.14
Normal	0.5	4	2.94	0.89	10.02	0.98
		8	2.91	0.79	8.78	1.03
	2	4	37.65	1.31	64.84	12.73
		8	24.36	1.16	43.64	9.05
Shifted	4	4	2.95	0.92	10.87	0.88

As the behaviour of the linear distribution is very similar to that of the accumulation approach, it showed little difference in the performance. Note that the sum of the distribution does not equal 1 when using the normal distribution algorithm. This can be seen especially for the higher factor, as the motions often ended up being short of the actual amount. Increasing the window size allowed closer representation of the full value, but the effects could only be observed with a large factor which did not improve the accuracy. The lower factored implementations showed attractive results due to the sharp drop-off which lead to very small amounts of merging between the frames. The quadratic approaches also showed improvement in the accuracy, especially with the higher factor. The effects of using a larger sized window did not show much of an effect when the factors were set larger, except for the latency issue described earlier. The difference between the shifted and the quadratic distribution were not significant, but allow the anticipation of the subsequent motion to improve the transition of the motion. The small decrease in the responsiveness is not a critical issue, but the approach can potentially be problematic if too much smoothing occurs from an incorrect factor size.

## 7.3 Motion model

### 7.3 Motion model

When converting from the tracker motions to the robot motion, various models can be applied to combine and determine the translation and rotation undergone by the robot. Since the fundamental base for local localisation is to precisely measure the instantaneous pose changes, the models being used must accurately portray the actual motion the robot has undertaken. This process typically involves the classification of the motion to be either pure translational or rotational motion, which leads to the derivation of the rotational pivot point, called the ICC. This approach allows any rotational motion to be simplified to a point and an angle, which can then be applied to the robot for that frame. One of the requirements to use this approach is a very fast frame rate, as all the motions between the frames are compressed into a single motion representation. As well as the camera having a high frame rate, the heavy and rigid body of the robot and the controlled set of contact points to the ground also assist in maintaining a consistent and smooth motion occurring between the frames.

Depending on the assumptions made for that frame along with the model being used, various constraints are placed on the robot motion and the type of errors that will be introduced. This is due to the non-continuous motion that is captured by the tracker, as well as any approximations made by smoothing algorithm introduced earlier. This often causes theoretically designed models to behave erratically, thus factors such as the robot's wheel configurations and typical motions it expects must be integrated with the sensor inputs for use in determining the motion.

#### 7.3.1 Exact motion model

Assuming that the motions measured by the trackers are precise and consistent throughout the frame captures, the center of curvature can be derived by identifying the line of possible location of the point for each of the trackers, then identifying the intersection point of the lines. The line itself can be derived by extending a perpendicular line from the midpoint of the motion vector, since the whole motion is assumed to be consistent and the symmetry constrains the motion. Using the derived point and the distance to the tracker location, the rotational angle can be determined. A special case is triggered when the two lines are parallel, which signifies a pure translational motion. Figure 7.8 below illustrates this concept.  $\theta$  represents the angle between the starting and ending location of the feature centered around the ICC.

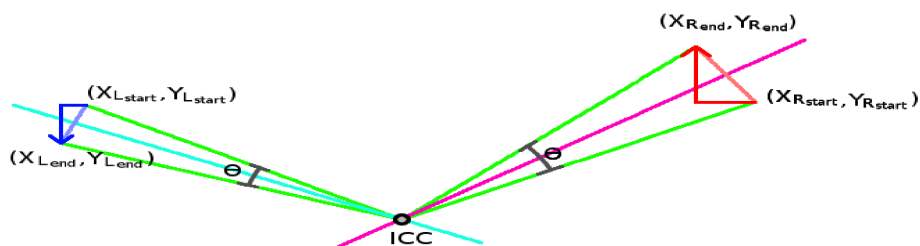


Figure 7.8: Exact motion model.

The precise motion detected by the tracker can be used to determine the ICC and the rotational angle. The arrows indicate the tracker motions.



### 7.3.1 Exact motion

The derivation of the pivot position for the exact motion model requires a number of steps using the starting and final positions of both trackers, starting with the line equation for the possible location of the pivot:

$$\text{Slope} = (Y_{\text{end}} - Y_{\text{start}}) / (X_{\text{end}} - X_{\text{start}}) \quad (32)$$

$$\text{Slope}_{\text{perpendicular}} = -1 / \text{Slope} \quad (33)$$

$$\text{Intercept} = ((Y_{\text{end}} + Y_{\text{start}}) - \text{Slope}_{\text{perpendicular}} (X_{\text{end}} + X_{\text{start}})) / 2 \quad (34)$$

Where  $X_{\text{start}}$ ,  $Y_{\text{start}}$ ,  $X_{\text{end}}$ , and  $Y_{\text{end}}$  represent the x and y positions of the initial and final tracker locations. The lines for the two trackers can be combined together to determine the center of curvature:

$$\text{Slope}_a * X_{\text{pivot}} + \text{Intercept}_a = \text{Slope}_b * X_{\text{pivot}} + \text{Intercept}_b \quad (35)$$

$$X_{\text{pivot}} = (\text{Intercept}_b - \text{Intercept}_a) / (\text{Slope}_a - \text{Slope}_b) \quad (36)$$

$$Y_{\text{pivot}} = \text{Slope}_a * X_{\text{pivot}} + \text{Intercept}_a \quad (37)$$

Where a and b represent each of the trackers. The derivation of the rotational angle can then be carried out:

$$\text{Radius} = \sqrt{((X_{\text{end}} - X_{\text{pivot}})^2 + (Y_{\text{end}} - Y_{\text{pivot}})^2)} \quad (38)$$

$$\sin(\Theta / 2) = \sqrt{(((X_{\text{end}} - X_{\text{start}})^2 + (Y_{\text{end}} - Y_{\text{start}})^2) / (2 * \text{Radius}))} \quad (39)$$

$$\Theta = 2 * \sin^{-1}(\sqrt{(((X_{\text{end}} - X_{\text{start}})^2 + (Y_{\text{end}} - Y_{\text{start}})^2) / (2 * \text{Radius}))}) \quad (40)$$

At this point, it is easy to see the inconsistencies that can be found, as the angles that are detected for the trackers may differ between each of the trackers. Since multiple parameters are known from the tracker motions, any excess attributes can be used as part of error detection and correction process which will be discussed later.

Although models that can derive the exact position of the pivot are often used in simulations, these approaches do not always translate well under realistic conditions, especially when the motions are measured with a finite precision and granularity. To reduce the errors, more sensors can be introduced to model the states of the environment and the robot, the precision of the sensors can be increased, or different motion models can be made use of to note any inconsistencies.

Although the performance of the tracking algorithm was quite reasonable when only considering the single dimension, the synchronisation and precision requirements for 2D motion meant that the motion vectors being combined often misrepresented the actual motion and resulted in an incorrect center of curvature being used.

### 7.3.2 Differential motion

Another model that is commonly used in simulations is a constraint based motion model, where the characteristic of the components that causes the robot to move is assumed to dictate all of the motion of the robot. This model is often used to model the forward and inverse kinematics to measure and anticipate the robot motion. The model simplifies the robot motion by disallowing any irregular motions, such as external forces, and assuming perfect and consistent behaviour of the moving components. For a differential drive system, the forward and backwards motions of

### 7.3.2 Differential motion

the two wheels are used to derive the center of curvature which has been constrained to a line joining the rotational axes of the wheels. Using the proportional motion of the two wheels and the placement of the wheels, the rotational point and the angle can be determined. Figure 7.9 illustrates the components of the approach and the derivation of the unknown parameters, where the blue arc represents the left wheel's motion while the red arc represents the right wheel's motion.  $L$  and  $R$  represent the motions detected by the left and right tracker, while  $D$  represents the distance between the trackers.

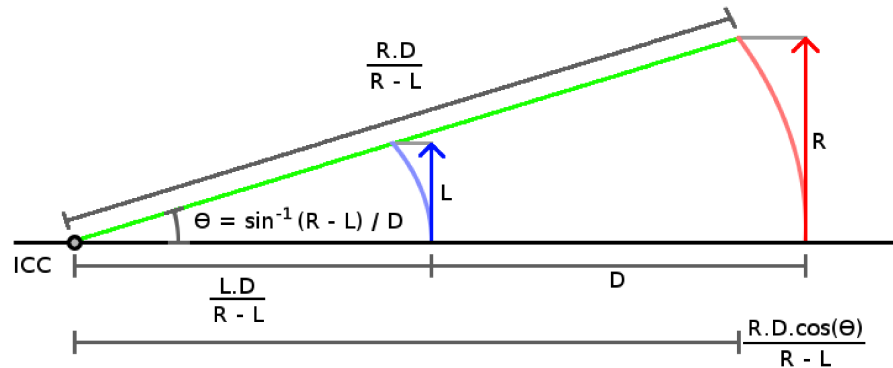


Figure 7.9: Motion model for differential drive systems.

The proportionality in the left and right motion vectors are used to derive the ICC and the angle of rotation. The arrows indicate the tracker motions.

The characteristics of this approach can be translated to the motions detected by the trackers when the cameras are mounted on the sides of the robot. Since the cameras provide feedback information about the robot motion, many of the problems with dead reckoning can be avoided while allowing simple calculations to derive the motion.

It is important for the tracker to be placed precisely, as the configuration of the camera with respect to the robot motion plays a significant role in the accuracy of the approach. This means that the trackers should be placed where they form symmetry along the rotational axes of the wheels. As the actual position can vary slightly depending on the surface and the wheels, the algorithm must assume certain constraints and expect certain levels of inaccuracies.

The placement of the viewing window for the feature was chosen to be slightly ahead of the rotational axis, as the majority of the robot motion would be in the forward direction, and does not allow dynamic adaptation to reduce the complexity and the false predictions. Due to the fast frame rate, the motions of the feature should remain quite small, thus any adaptive behaviour will have little benefit to the overall precision. Even if an algorithm was able to determine the ideal location for the tracker, the uniqueness of the ground texture below will modify the position of the tracker to improve the feature tracking process.

The inaccuracies in the measurements lead to selecting the incorrect position for the center of curvature, but the constraints placed by the model restricts the placement of the rotational point, thus forces a much smoother motion. However, if

### 7.3.2 Differential motion

the assumptions made by the model is invalidated, such as when the robot slips, the model will force the irregular motion to be mapped to the constrained motion, thus will register an incorrect motion. Although the inherent error tolerance is welcomed when the reduced precision in the measurement does not allow the correct motion to be captured; the severe constraint does not always allow realistic motion to be registered.

Depending on how the 2D tracker motion is made use of, different variations of the differential motion model can be implemented. The first of these models combines the longitudinal and latitudinal motions as the arc length, thus allows for a simple translation to the original approach. It is possible to combine the two displacement values as an angled motion, or a simple addition of the magnitudes. An alternate model is to assume that the longitudinal motions are simply the by-product of the rotation, and only the latitudinal motion is used to determine the motion of the tracker. The results of these models can be seen in table 7.7, where no smoothing algorithms were used.

Table 7.7: Accuracy of differential motion models.

Tracker motion	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
Arc length	3.21	2.93	13.56	3.22
Latitude + Longitude	11.32	5.31	29.66	5.82
Longitude	2.93	1.31	10.2	1.31

As the above shows, the use of just the longitude allowed for much more attractive results where the subtle latitudinal motions are ignored. The stability of the robot in typical operating surfaces means the approach is an attractive alternative to the exact motion model if motions by external forces do not apply.

### 7.3.3 Hybrid motion

To overcome the weaknesses of the two models above, a hybrid motion model is introduced to determine and switch between the appropriate model to suit the motion detected by the tracker. This requires the detailed analysis of the motion behaviour of the robot in operation.

The constraint based approach provides a reasonably accurate model for the majority of the robot's motion when under normal operation. Using this as the basis, the situations where the robot performs an irregular motion can be detected and used to trigger an alternate motion model. Since the tracker is capable of tracking motion in multiple axes, the proportionality between the motions in each axis can be used.

Under the assumption that the wheels are the sole provider of the motion, any sideward motion will be accompanied by a larger forward or backwards motion. The proportionality between the two depends on the motion measured by the trackers, as well as the distance between them. With the tracker placed to form symmetry around the rotational axis, the latitudinal motion will be maximised halfway during the motion and cancels out by the time the motion is registered. To remove this effect of

### 7.3.3 Hybrid motion

latitudinal motion cancellation, the tracker must be placed such that it does not cross the rotational axis. This meant that the starting position for the tracker has to be placed on or away from the rotational axis and to allow the motion away from it. Since the maximum displacement of the tracker with respect to the rotational axis depends on many external factors, using a pre-empted tracker motion to position the starting location is a difficult problem. The initial tracker location is thus simply placed directly on the wheel's rotational axis, such that any natural motion will cause the tracker to move away from the axis.

To determine the maximum displacement, the slope of the function can be used. This indicates that the latitudinal motion is increased as the longitudinal motion is increased because of the high frame rate, which restricts the maximum rotation that can be observed. When two tracker motions are involved, the interaction between them must be considered. Figure 7.10 illustrates the latitudinal motion for various longitudinal motion pairs.

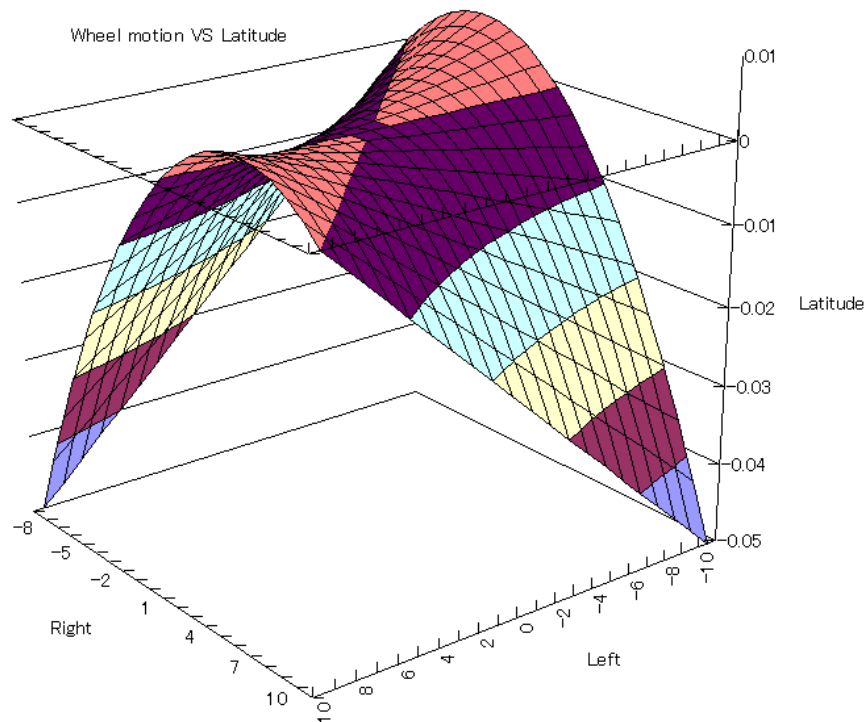


Figure 7.10: Latitudinal motion from two tracker motions. The relationship between the latitudinal motion given the left and right wheel motion.

The figure above clearly shows that the maximum latitudinal motion occurs when both sides travel at the maximum speed in opposite directions. Although it is possible to derive the exact expected latitudinal motion values for a given longitudinal motion, the granularity and errors in the measurements often leads to slight inconsistencies between the two directions. The effect of this is reduced from the smoothing of the motions introduced above, but is not reliable enough to confidently say that the proportional motion becomes accurate. Instead, a simple lookup table can be used to determine the expected value, which includes an error tolerance value to assist in determining the correct proportionality. As the current implementation

### 7.3.3 Hybrid motion

limits the maximum longitudinal motion to be 10 pixels in either direction, the table is given a capacity of 100 entries, where each entry is mapped to a 2 by 2 pixel interval.

Based on the fitting with the table entries, an alternate motion model can be triggered to process the irregular motion. This type of motion is unlikely to occur as part of a continuous motion due to the friction caused by the wheels, but instead, be the result of small bumps or slips the robot may experience. Although the tracker is reliable during normal operation, sudden and irregular motions can throw off the predictions, blur the image, and invalidate some assumptions with regards to the allowed motions. This means that the use of the exact motion model is more likely to make use of inaccurate tracker motions, which can introduce large amounts of errors to the pose.

Instead of modelling the motion as a rotation, it is possible to simplify the motion as a pure translational motion involving the averaging of the two motion vectors that are detected. The validity of this approach depends greatly on the similarity of the detected motion vectors and the likelihood of this type of motion occurring within the environment. As well as the likely causes mentioned above, the current set of sensors and the operational environment allows for slightly easier recovery from angular errors than displacement errors as the camera sensors can provide a higher precision than the range finders when used to determine the relationship against the environment.

The testing of the hybrid motion was quite difficult to compare with the other tests, since it had to test for the accurate registration of irregular motions. Instead of artificially constructing an experiment that just involved the irregular motions, the test involved the same configuration as the previous test to compare their overall performance against earlier implementations. The results of the tests can be seen in table 7.8.

Table 7.8: Accuracy of the hybrid motion model.

Hybrid motion	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
Differential + exact	2.98	1.83	16.42	6.9
Differential + translation	2.44	1.18	5.32	0.93

The experiment showed that the use of the exact motion model did not allow the accurate portrayal of rotational motions due to the greater possibility in the center of curvature, as well as the dependency on synchronising the motions detected between the two trackers. The switching of the model to a translation model allowed for a much more precise localisation algorithm, simply by specifying a threshold value to distinguish the different types of motion. Although this is dependant on the surface types and the types of motions to be encountered by the robot, it shows promising results in providing the local pose.

As a final comparison, three motion models are compared using the windowed smoothing algorithm with a quadratic function of factor 4 and size of 4. This is summarised in table 7.9.

### 7.3.3 Hybrid motion

Table 7.9: Motion model precision using smoothing.

Motion model	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
Exact	2.88	1.21	88.12	9.2
Differential	2.88	0.78	9.72	0.97
Hybrid motion	1.95	0.41	2.32	0.53

## 7.4 Error handling

To help prevent the localisation algorithm from including erroneous information, the motions that are detected must be analysed and adjusted before being applied to the pose. This process consists of two main components; the detection and the correction of errors. The approaches that are investigated are based on local information rather than the long termed global error reduction, which allows the motions to be analysed quickly to reduce the cascading effect the error may cause. It also decouples the issue to just the localisation module, but can allow separate error correction processes later on by a different module.

### 7.4.1 Detection

As noted earlier, the detection of errors can occur at various stages when analysing the tracker measurements. They are often determined by comparisons against known constraints and domain knowledge to flag suspicious measurements. It is also important to note the difference between an error which can potentially do significant harm and those that are uncontrollable as they are inherent in the approaches or the devices used.

The use of the exact motion model allowed for some extra parameters to solve for the center of curvature. It is possible to derive the pivot position using the other parameters using alternate sets of equations, which can then be used to compare the consistency. Due to the precision in the tracker measurements, there will be small amounts of differences which will have to be distinguished amongst larger inconsistencies. The two simple checks that can be made are the angular differences detected from the pivot point and the consistency in the distance between the tracker positions. Since the robot and the camera remains rigid, the distance between the starting positions and the final position where the features were tracked to should remain the same.

Knowing the precision limitations of the trackers, it is possible to derive a region for where the center of curvature may reside. Given that the sub-pixel motion may not be registered, the tracker measurements may be, at most, 1 pixel different to the actual motion. With the potential pivot positions established, the motion can be validated against expected motions with more leniency than against a single value. By adding the tolerance adjustments to the final positions, the range of positions for the x and y values can be determined.

### 7.4.1 Detection

$$\text{Slope} = - (X_{\text{end}} + X_{\text{tolerance}} - X_{\text{start}}) / (Y_{\text{end}} + Y_{\text{tolerance}} - Y_{\text{start}}) \quad (41)$$

$$\text{Intercept} = ((Y_{\text{end}} + Y_{\text{tolerance}} + Y_{\text{start}}) - \text{Slope} * (X_{\text{end}} + X_{\text{tolerance}} + X_{\text{start}})) / 2 \quad (42)$$

Similarly, the adjusted positions for the constrained motion can be derived.

$$\text{Distance} = D_{\text{tracker}} * (D_R + R_{\text{tolerance}}) / ((D_L + L_{\text{tolerance}}) - (D_R + R_{\text{tolerance}})) \quad (43)$$

where  $X_{\text{start}}$ ,  $X_{\text{end}}$ ,  $Y_{\text{start}}$  and  $Y_{\text{end}}$  refers to the start and end position for X and Y coordinate points,  $X_{\text{tolerance}}$ ,  $Y_{\text{tolerance}}$ ,  $R_{\text{tolerance}}$  and  $L_{\text{tolerance}}$  refers to the variations allowed in the position and measurements,  $D_{\text{tracker}}$  is the distance between the trackers, while  $D_R$  and  $D_L$  represent the longitudinal motions detected by the right and left trackers. Note that the difference in the slopes of the lines used to find the center of curvature has a significant impact on the range of possible locations, thus the benefits of identifying a region for acceptable pivot positions can be very limited, especially when the robot does not rotate.

Another way to check for errors using a threshold value is at the tracker level, where the correlation score can be used to determine whether the match meets the expected score. Since the correlation scores are dependant on the type of ground textures, it is important to modify the threshold to suit the environment. This can be implemented by techniques such as monitoring the ground textures or maintaining the recent correlation scores to observe the trends in the scores.

Although the above approach may have potential, there is no guarantee that the expected score appropriate for the current ground texture. This meant any fluctuation in the scores, such as when sub-pixel motions occur, would flag the motion as erroneous and would frequently require a separate validation algorithm.

#### 7.4.1.1 Redundancy

The error detection based on constraints and domain knowledge can be quite effective at times, but it also means a portion of the measurement's range is consumed by sentinel values. Another limitation of the approach is that the effectiveness of the error detection depends greatly on adhering to the constraints, which cannot always be guaranteed, and also in knowing the domain beforehand. A commonly used work around for this is to make use of redundant measurements which can be used to measure the consistency for both the detection and correction of errors.

Since the tracker is implemented in software, introducing more trackers does not require significant changes as most of the issues, such as synchronisation, consistency and placement, have already been considered. The decision in selecting the location is determined by what type of information is required. As mentioned earlier, the more distant the tracker is to the others, the more unique the detected motions are, thus a decision must be made between implementing a more independent tracker or simply providing a backup for the existing tracker. Using a dedicated backup for an existing tracker can simplify its purpose by focusing on consistency thus can be implemented more efficiently than an implementation with extra independent trackers. Table 7.10 shows the extra processing time consumed from introducing one or two more trackers for a single camera. Note that this only considers the tracking of the features and do not implement any other algorithms that

### 7.4.1.1 Redundancy

may be used to remove the errors.

Table 7.10: Processing time for extra trackers.

Trackers	Initialisation (ms)	Tracking (ms)
2	0.11	0.07
3	0.2	0.16

Due to the sophisticated algorithms that are implemented for the tracker, each additional tracker increases the overall processing load by a significant amount. This increase can potentially cause delays in the other timing crucial processes, perhaps even delaying the capturing of the next frame and causing a whole frame to be skipped. This can have disastrous effect on the localisation algorithm, as the incorrect tracking effects the feedback information used to anticipate future motions. To reduce the overheads in using the extra trackers, techniques such as sharing of common or similar attributes between trackers and triggering the extra tracker on demand could also be considered.

### 7.4.2 Recovery

The second part to the error reduction process is the correction of the detected errors. This phase involves the use of various approaches and the available information to reconstruct the intended measurements. The types of information that can be used include the measurements from adjacent time frames, predictions from trends and constraints, as well as measurements from the redundant trackers.

On some occasions, it may also be possible to use the erroneous measurement as a guide to identify the cause of the problem to adjust the recovery process appropriately. Characteristics such as a sudden decrease in the overall intensity fluctuations, which is likely to be caused by motion blur, or the motion being reasonably consistent but was voted as being the outlier, which may be caused by the precision errors, indicate that the erroneous measurement does not need to be discarded completely.

#### 7.4.2.1 Merge

If there are redundant measurements available, the information can be merged together into a single value. This can be based on simple averaging or weighted interpolation, depending on any confidence values that are available. This, of course, requires that the values being used have been filtered to remove the erroneous values and are reasonably similar to each other. It is important to consider the sources of the information before being combined, as its origin determines how the value needs to be converted before they can be combined.

The effects of merging adjacently placed trackers can be seen in table 7.11, which showed some improvement in the accuracy for a translation, and reasonable results for rotation. This is likely due to the infrequent occurrence of erroneous measurements, thus introducing more smoothing to the motion observed by the camera.



### 7.4.2.1 Merge

Table 7.11: Merging of redundant tracker motions.

Trackers	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
2	1.96	0.56	2.98	0.73
3	1.77	0.21	3.43	0.9

Although merging the motions showed attractive results, it can potentially combine erroneous tracking in the absence of a filter to remove the problematic reading first. In this particular case, the redundant tracker was placed close to the original tracker, thus the tracker motions did not require much alteration before they were combined.

### 7.4.2.2 Eliminate

If the measurement is deemed erroneous and cannot be used to derive the intended motion, the motion must be discarded and recovered by other means. Since many of the recovery approaches make use of the historical measurements, it is desirable that consecutive measurements are available and be accurate. Although this is difficult to enforce, it is possible to flag the current localisation state as being unstable to inform the other modules to make appropriate adjustments, such that measures can be taken to avoid using the current pose. This could include slowing the robot down or reversing, so that the landmarks can be observed again to correct the pose retrospectively.

Depending on the consistency of motion and the availability of historical information, it may be possible to predict the current position. Although the idea seems reasonable, the erroneous measurement is typically caused by irregular motion, thus would not fit the anticipated motion pattern. However, this approach does give an instantaneous result if the pose is required immediately.

Combining the newly observed motion and the previously observed motion can often allow for a better approximation of the motion in between. This approach presumes that the measurements surrounding the erroneous one are valid and the motion is smooth. If consecutive tracking are deemed erroneous, the predictions can be further extended and retrospectively applied for a longer period. This places a greater emphasis on the presence of smooth motions and the accuracy of the measurements, but may be necessary due to the slow response time between informing the other modules to make adjustments to reduce the error causing behaviour and in case the irregular motions, such as bumps, carry on for multiple frames. Table 7.12 summarises the accuracy of implementing the prediction and retrospective error correction approach.

Table 7.12: Accuracy of prediction and retrospective error correction.

	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
Retrospective	5.44	1.3	13.31	6.91

### 7.4.2.2 Eliminate

Although the motion of the robot seemed smooth and consistent, the motions being tracked often fluctuated, especially when rotations occurred. This meant the inter-frame motions are not as consistent, as confirmed by the findings in the precision algorithm in chapter 6.

A typical use of the redundant trackers is to implement a voting mechanism. The tracker measurements can be compared to identify any inconsistent measurements, which can then be flagged for subsequent error correction mechanism. In order to implement a voting algorithm, there must be an odd number of measurements, or a weighting attached to the measurements to allow disambiguation. However, if the measurements are significantly different, such as with the case when the trackers are placed far apart, the approach is less effective due to the increased range of plausible motions and the wider variety in the tracking.

A very simple implementation of this is to introduce two more trackers for each of the original trackers, thus allowing a local voting mechanism before they are combined. Since the positions of the trackers must differ, it is important to place the extra trackers appropriately, such that the motion characteristics can be calculated easily. One strategy to achieve this is to place the three trackers along the same axis as the wheel rotation and vary the distance measure to the center of the robot. The voting algorithms that were implemented compared the displacement values and noted the outlier, followed by either the selection of the median value or the averaging of the non-outlier motion vectors. The results of this can be seen in table 7.13. The hybrid motion model with the windowed smoothing algorithm, as per the best performing algorithm was used for this experiment.

Table 7.13: Results of voting off outlier

	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
Median	2.52	0.37	39.04	0.51
Average	2.42	0.27	46.93	0.38

An interesting observation which was made after implementing the voting algorithm was that the type of motion contributed greatly to the performance of the algorithm. It was able to remove the outlier tracker reading very effectively when errors were introduced, but the majority of the rotational motions were also deemed erroneous. The subtle changes were only observed after they had accumulated enough sub-pixel motion, which occurred at different times between the trackers. Instead of blindly removing the extreme values, it would be beneficial to apply a threshold value to correctly classify an erroneous reading before they are discarded.

## 7.5 Practical considerations

While consideration to the workings of the localisation algorithm has been investigated in detail, several other important issues must be addressed concurrently for the approach to be of practical use. Some of these issues are concerned with how the algorithm is affected by the environment, while some deal with the internal configuration issues that may require adjustments.

## 7.5.1 Motion

### 7.5.1 Motion

When transforming the sensor measurements to robot motion, it is important to establish a well placed reference point for the coordinates. As briefly mentioned in chapters 4 and 6, the information from the sensor must be mapped to a uniform coordinate system to be shared by other modules. The coordinate space joining the multiple sensors is typically set relative to the robot and does not have a strong preference for a unit to be used. When the robot's motion is mapped against the environment, it must be bound to a certain coordinate system, whether it be something the observers of the robot can understand or practical for internal representation. For this reason, the coordinate system used by the robot typically uses the observer's preference of units to reduce the conversions when the pose is viewed.

Defining the coordinate axes for the environment can be a difficult task, as the robot can be activated in an arbitrary position and there are rarely any indicators which specify the ideal starting location. When the robot's pose is being combined with external information, such as when data from multiple activations are being compiled or if global pose indicators such as a compass or GPS are used, the reference point becomes more crucial. However, in terms of local localisation, the coordinate axis can simply be placed at some arbitrary point, such as the starting location and use the relative coordinate points from then on.

When using only the downwards pointing cameras, global localisation is a very difficult task to achieve due to the lack of long term features. Overcoming this requires the use of the other sensors to help disambiguate between the similar looking locations, or a distinguishable texture pattern to be present on the ground, such as the change in the floor material or significant markings like stains or cracks. This is investigated in more details in chapter 11.

Since the localisation algorithm is highly dependant on the precision of the calibration measurements, even the slightest deviations from the actual value causes an accumulation of errors, which is difficult to correct in normal situations. This is often because the robot does not back track the same path and the error in the scale is accumulated. This can be seen from the inconsistency between the accuracies of the forward and backwards values in the experiments.

To minimise this issue, the calibration and testing processes must be repeated numerous times while being careful not to alter the placement of the cameras. This process requires significant manual intervention, but is unavoidable without a secure placement for the cameras or an automated calibration phase during the initialisation of the robot.

Since this primarily causes scaling errors, it is also possible to counteract this by maintaining the motion vectors in memory and applying the properly scaled version if and when calibration data becomes available, possibly from long term landmarks or external pose indicators. This allows the robot to avoid the initial set up time by treating the measured motions as proportional values instead. Similarly, the robot can be encouraged to reverse its motions instead of rotating and making forward traversals all the time. This will allow cancellation of scaling errors when the robot returns to the original location.

## 7.5.2 Surfaces

### 7.5.2 Surfaces

Since the texture of the floor during the experiment was fixed, the algorithm must be validated on other surfaces the robot could encounter. The current configuration of the robot assumes operation in indoor environments, but the tests also included some surface types that were typically seen only outdoors.

The range of surfaces that were tested included those with different reflectivity, regular and irregular patterns, as well as different levels brightness. To avoid damaging the robot, surfaces such as grass and sand were not conducted. The large variance in the height would not have met the pre-conditions required for the algorithm, so the results would have been very inconsistent. Measurements taken on rough surfaces, like brick and concrete included minor adjustments to the robot so that the balls used as caster wheels were not scratched. This involved placing a plastic sheet under the caster wheels to roll on, which were away from the camera's views and the path of the foam wheels. These meant some of the bumps it would have normally encountered were avoided, but since the robot is not intended to operate on these types of surfaces, the validity of the visual odometry can still be measured from the perspective of a different looking surface texture.




The testing procedures were kept as the same, where the robot would perform a forward and backwards traversal test, followed by a rotational motion test. The various settings on the robot, such as the camera position, lighting and tracker parameters were kept consistent, while the algorithm that was used was a hybrid motion model with a windowed quadratic smoothing factor of 4, with the size also being 4. As for the error correction algorithm, two trackers per camera were used to allow merging if the longitudinal motions were within 2 pixels of each other. If there were inconsistencies between the redundant trackers, the tracker with the better correlation score was used. Table 7.14 summarises the results.

The performance of the visual odometry algorithm showed reasonable consistency on many other surface types, including those with very little visible differences such as the table top and tiled floors. The small bumps and grooves on these surfaces provided the uniqueness to be able to distinguish them locally. The textures on the vinyl floors consisted of many small marks, such as dirt and scratch marks, which allowed very distinctive features to be present.

The localisation on timber floors and rubber mats provided interesting results, where the patterns were quite visible to humans, but the tracking algorithm was unable to distinguish them due to the difference in scales to what we recognise as patterns to what the camera can see. The patterns that were present were too large and repetitive, which caused issues with similar correlation scores being evaluated for different features. A possible strategy to avoid these problems is to increase the feature sizes dynamically based on the uniqueness scores of the ground textures, to control the anticipated motion of the feature by slowing down the robot, or by using a camera with a much faster capture rate such that the repeated pattern is not observable within the search area.

## 7.5.2 Surfaces

Table 7.14: Localisation accuracy on different surfaces.

Surface	Sample	Translation (%)		Rotation (%)	
		Forward	Backward	Clockwise	Anti-clockwise
Vinyl		1.87	0.29	2.83	0.58
Table		2.37	0.52	3.08	0.92
Timber		8.21	1.07	7.18	3.84
Rubber		17.3	0.28	18.71	6.17
Tile		1.94	0.2	2.96	0.86
Brick		1.7	0.42	3.21	1.6
Concrete		2.44	0.27	2.69	0.76

Overall, the proposed localisation algorithm performed well on many of the typical textures that an indoor operating robot would encounter. However, operations

## 7.5.2 Surfaces

on surfaces with large and repetitive textures must require adjustments to the feature selection process to cater for the differences in the texture patterns.

## 7.5.3 Extended traversals

The last and the most important test conducted was the performance of the approach over a long period of time. The algorithm was compared against manual measurements and dead reckoning algorithms over a traversal around the lab environment, with position measurements taken and motion commands being given every 500 mm of motion. Slight adjustments to the ideal motion was necessary to avoid collisions with the office furniture, due to the slight inconsistencies in the wheel motions, such as the dimensions and speed, which caused the robot to stray off to one side. This is evident in the arced path taken by the robot, as seen in figure 7.11. Some of the corrections were over-applied to allow the robot to reach the desired points after the arced motion by guessing the required increase to the rotation. The result of this experiment was averaged out over three runs.

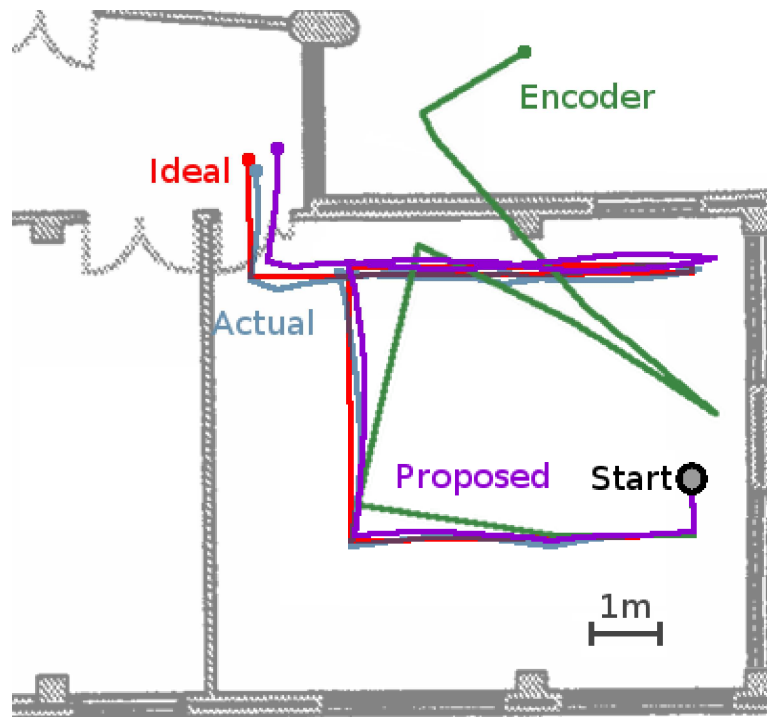


Figure 7.11: Comparison of extended traversals.  
The colour coded paths illustrates the dramatic improvement in the localisation implementation using only local localisation.

The traversal covered a total distance of 26.5m, where the difference between the encoder based localisation and the proposed visual odometry technique can be observed quite distinctively. Although the errors are accumulative, the difference between the actual motion and those observed by the localisation algorithm is quite small, making it an effective algorithm to be implemented for local pose maintenance. With the proposed algorithm, it is possible to perform the global localisation more efficiently and less frequently, as the measurements do not stray too much to avoid the corrections from unrealistic or ambiguous poses.

## 7.6 Summary

### 7.6 Summary

The trackers that forms the foundation for the localisation algorithm performs efficiently and effectively in registering translational motion, but is unable to detect rotations due to the constraints placed by the configuration and the algorithm. The translations that are detected by multiple trackers can instead be combined to identify the robot's pose through the extra constraint introduced between the placements of the trackers.

The introduction of multiple trackers considers additional issues to correctly synchronise the multiple motion vectors and uses an appropriate motion model to convert the tracker motion to robot motion. The placement of the extra trackers had to consider the distance between each other to register distinct motions. This allowed the narrowing of the possible motions due to the imprecise motions that could be observed from the presence of sub-pixel motions. Instead of being restricted to a small viewing area, additional cameras are included to increase the flexibility in the tracker locations. In doing so, different motion models are introduced which allowed more control over how the tracker motions related to the robot motion.

To encourage the smooth transition in the motions for better synchronisation between the trackers, various algorithms were integrated to account for the granularity in the motion. It was found that using a window to view the previous frames with a strong weight to the more recent motion allowed for improved accuracy from the blending of the motion vectors.

To improve the effectiveness of the localisation algorithm, error detection and correction mechanisms were also incorporated into the algorithm. The use of threshold values and redundant trackers allowed for some improvement in the overall accuracy, but issues with resource usage must be addressed to note the overall gain in the accuracy if an alternate error correction approach becomes available.

With the visual odometry technique implemented and tested for applicability for the mobile robot, a reliable local pose has been made available for the other modules. The proposed technique minimised the dependency with other modules, thus can be improved quite easily with additional sensors and algorithms developed in the future. This will include an algorithm to map the local pose to a global pose.

## Section 3 – Mapping

*“The derivation of a useful map requires both direct and indirect sensory information thus requires a rapid cycle of refinement.”*

The second half of the SLAM paradigm focuses on the generation and usage of the environmental map, which comprises of a virtual representation of the environment that links to the physical world. This allows the grounding of the localisation procedure as it assists with the global localisation problem (Olson, 2000), as well as introducing a historical database to be constructed with regards to the state of the surroundings. The representations can range from modelling the physical properties of the environment, such as the location, and texture, to higher level representations, such as classifying a region of space as an office or a corridor.

Whether the maps are provided beforehand or not, the mobile robot must make use of the available sensors in order to perceive the characteristics of the environment. The various sensors and systems provide different ways to classify the immediate surroundings, which can later be combined as part of a larger map that extends beyond the robot's current view. As each of the sensors measure certain attributes in various ways, they require appropriate translation processes when being merged. This requires careful calibration of each sensor, as well as appropriately selected algorithms to model the interaction of the sensor to the environment.

Although some sensor characteristics can be determined during the calibration stages, they are often dependant on the actual environment and cannot be completely anticipated. The fluctuation in the measurements caused by these are often tackled by limiting the operational ranges or combining multiple sensor measurements for disambiguation (Dudek et al., 1995), which can occur at various stages before the information is represented in the map. The inaccuracies from the sensor readings mean the attributes are often applied as a probability of a particular state (Thrun et al., 1998).

The attributes that are detected allows the robot to make decisions based on the surroundings, as well as being able to relate the features to known objects. As many of the sensors use a specific modality, many of the comparisons can only be achieved between the same sensors. There is also another issue of timing and the visibility range, which can limit the interactions between the sensors. By converting the sensor readings to that which can be related to another sensor's readings, the sensor measurements are able to influence each other to build up a more informed representation of the environment.

Storing the attributes, which include any derived information, involves arranging the information in an easy to access and modifiable data structure, as well as placing constraints on the range and amount of information, typically by using compression mechanisms. These implementation issues play a crucial role in real time operations, especially in the presence of high volumes of data. Although key attributes can be filtered by their respective sensor modules, combining the information to a uniform representation requires carefully designed algorithms and a significant processing



time.

One of the key aspects of SLAM is that the localisation module influences the mapping module, while the mapping module provides feedback information for the localisation module. The close coupling between the two means constant cross-talks between the modules occur, which is aimed at gradually reducing the uncertainty in the pose and the map. However, the cyclic dependency can also lead to gradual divergence over time. To overcome this issue, many strategies are required to anticipate, detect, and correct any errors that may propagate by means of calibrating dynamically.

This section deals with the map construction techniques implemented on the mobile robot and covers the issues ranging from efficient map usage, merging of sensor readings, as well as the detection and handling of land marks and dynamic objects within the scene (Coombs & Brown, 1993; Fox et al., 1998 (a); Stachniss & Burgard, 2005). Chapter 8 introduces the various mapping issues and the types of maps that are available, while chapter 9 introduces several map construction techniques using the range finders on the robot. Chapter 10 investigates the use of a forward looking camera to gather the scene information using a different type of sensor to the range finders. Finally, the derivation and integration of higher level concepts are described in chapter 11.

## Chapter 8 – Representing the environment

The portraying of the environment results from the compilation of the instantaneous sensory information into a consistent representation. As well as being able to provide relational information between the sensor readings, the map can allow higher level constructs to be derived and provide historical information for subsequent sensor measurements to compare against. These benefits contribute to the successful and accurate operation of the mobile robot, as most tasks require more than just the instantaneous sensory information (Bailey, 2002; Thrun, 2002).

From the perspective of the representation, the map is required to store the sensory and derived information with enough detail, such that the tasks being carried out can operate with some certainty and confidence. Depending on the types of interactions required to the map, the efficient accessibility and accuracy must be maintained (Kuipers & Levitt, 1988). This is dependant on the data structures being used to store the attributes, as well as the algorithms to interface between the sensor readings and the map. The third issue to be considered is the organisation of the maps, such that the environment can be viewed from multiple perspectives. The isolation of different attributes creates a more cluttered representation, due to some repeated information, but allows for easy distinctions between the attributes and leaves scope for different map management algorithms in the future.

### 8.1 Map generation

When sensing the environment, the robot may be exposed to several different forms of input, which range from direct sensor readings to pre-generated maps by other systems. Handling the different forms of input and the unique information they represent requires various techniques to effectively process and use the information. This is due to the semantic information (Hild, 2000; Kuipers & Byun, 1991), capture time, and the accuracy of the information often dictates how they are processed.

#### 8.1.1 Attributes

In terms of the content of the maps, the types of attributes that are commonly stored include the likes of the location, occupancy, texture, elevation, and measures of interestingness, which are typically derived from analysing the features within the map. Since the addition of attributes incurs extra processing and memory costs, only those that are necessary for the particular task is calculated and maintained.

The most commonly used attribute to be assigned to the entries in the map is the location of the interesting point. Typically, this is assigned from the relative pose based on the robot's current pose. However, in some cases, this attribute is registered as relative positions against landmarks and do not have a numeric value assigned to it.

Another commonly used attribute by mobile robots is the occupancy of an area.

### 8.1.1 Attributes

This is crucial to the functionality to a mobile robot, as it indicates both the areas it can and cannot explore, and indirectly, the areas it has and has not explored. A wide range of devices that are mounted on the robot allows for this measure, such as range finders or even just the pose information by the robot. The occupancy also allows the other attributes to be attached using the positions and the semantics of the sensor readings.

An interesting attribute that is sometimes used is for the distinction between individual objects within the scene. Although this attribute requires complex correlation process between the sensor readings or prior knowledge about the shape of the objects, this can allow for inter-object and dynamic properties to be derived, together with the ability to focus on a particular object of interest. This ability is crucial in environments where objects are constantly moving or when the robot is required to search or identify a specific object. This may include the detection of doors and people that move while the robot is observing the object or when the robot returns to note the change in the occupancy of the area.

Each of the sensors exhibit varying behaviours depending on the environment and the level of noise and ambiguity expected for the device. For this reason, it is often desirable to assign a weighted value or a probabilistic model to indicate the confidence in the assignment of the attributes to the map (Basye et al., 1989). This value may include the error tolerance of the sensor or the detection of inconsistencies, such as those derived from correlation scores between the sensor reading and one that is expected from the stored map. The variation in the confidence allows the scope for various disambiguation techniques using historical information and alternate perspectives from multiple sensors.

When converting analogue data to a digital representation, there is a limitation to the level of precision that can be achieved. The limit to the precision is dependant on the sampling capability of the sensor, the resource requirements, and also the precision requirements for the specific application. It is important to note the precision the map uses with respect to the specific tasks, as this can introduce the misalignment between the environment and the representation, similar to the previously encountered issue with aliasing errors.

### 8.1.2 Sensor inputs

The most direct input to the robot can be made using the on board sensors to perceive the current state of the immediate surroundings. The characteristics of each sensor determine what information can be derived about the environment, as well as how it will affect the map. In some cases, each of the sensor readings must also be time-stamped due to latency issues or delays introduced by batch processing the sensor inputs.

The communication between the sensor and the map often involves filters being applied very early on to reduce the unnecessary components and convert the readings to a more compact and useful form. This also assists in the isolation between the different modules, as they can each parse and extract the relevant information. Although some redundancy may appear, it is possible to reduce the processing requirements by carefully applying the filters in a sequence, such that many of the

### 8.1.2 Sensor inputs

processed data is reused. Other strategies used to improve the efficiency include the use of simple shapes and templates when interacting with the data, the use of constraints by knowing the range of possibilities beforehand, and setting the appropriate update rate of the map, as changes to the environment often does not occur as fast as the sampling rate of most sensors.

When selecting the type of sensors to use, the usability of the modality, the characteristics of the information it captures, and the reliability plays a significant role in terms of effectively modelling the environment. The sensors must capture the scene through various environmental conditions, which may require interactions with multiple sensors to accurately interpret the sensor readings. The modality dependant behaviours are sometimes neglected due to the small amounts of effect it has, or the lack of sensors to be able to distinguish the changes in the environmental condition. As many of the applications for mobile robots have specific tasks to be carried out, the range of operating environment are often constrained, thus the adaptive behaviour can be removed. This is a common and reasonable assumption to be used, since many experimental platforms are focused on developing a small set of algorithms to solve a very specific problem instead of deriving a general purpose algorithm for all situations. That said, many of the environmental effects can be modelled through simple scaling, which can be easy implemented if extra sensor for detecting the change in conditions are available.

Since many of the sensors are included with the intention of actively sensing the conditions of the environment, the particular modality they use is restricted to those that are continuously available when desired. This defines the attributes that can be determined, but at the same time, the selection of the appropriate attributes also depends on the type of map being constructed and the requirements for the map. In a typical case, the unnecessary attributes are derived information from the sensors, which simply means they are not produced if they are not required. This is due to the careful planning used before the sensor is integrated, as each device incurs a cost in both material and processing wise.

Although it is possible to simply capture the state of the scene according to the sensor readings, it is often desirable to assign a measure of importance or interestingness to various portions of the map after analysing the other attributes of the map. Although these values can require significant processing time to derive and may not be consistent from different perspectives, they can be used to identify landmarks and correlation points for localisation and to uniquely classify the area.

### 8.1.3 External systems

Using the sensors in the natural environment can lead to many unforeseeable problems that may hinder their performance. These may include issues like changes in lighting or difference in the reflectivity to the calibration data. The problem is enhanced when there are no references to identify the occurrence of these events to modify the sensor usage.

One strategy to overcome this is to interact with external systems which can inform the robot with consistent or grounded measurements. A simple example of this is a calibration feature that is used to reset the sensor's parameters, but can also

### 8.1.3 External systems

extend to pre-defined targets such as bar codes with encoded information. These calibration markers are commonly seen in environments where constant errors to the sensor readings occur, or when the computational load must be kept down. Often times, this involves careful crafting and placement of the markers and the integration of the associated algorithm to identify and correct the current settings using the difference measure determined from the pose or data from the markers.

When templates are used, the variations that commonly occur are usually not included as part of the constraint. An example of this may be a template with a specific colour or shape, but not both. This allows some flexibility in the attribute, which may be slightly different to the expected due to the conditions of the environment. The variations that are allowed introduces the ability to encode independent information into the marker, such as those seen in road signs, where the shape and the positioning of the sign is matched by the template and the arrangement of the colours on the sign depicts a particular information using a separate template. This idea is commonly seen in controlled lab environments, where encoded markers are placed around the environment to inform the location of the robot.

A slightly different approach that can be used is the exchange of information from externally located sensors or systems that can observe the state of the robot or the environment. These include systems like the GPS, surveillance cameras, or another robot that is operating concurrently (McLurkin, 2004). These allow dynamic and adaptive information to be sent to the robot, sometimes from a fixed pose, and can often be considered as another sensor measurement. These external systems are typically equipped with multiple sensors, as well as several derived attributes of their own, which can be passed onto the robot to allow multiple perspectives of the environment without physically moving around.

Although the set of constraints placed on the external sensors usually allow confident and accurate measures to be made, the dependency to the other system can limit the applicability, especially since the other system must be placed in the appropriate place before the robot can make use of them. The use of the calibration marker is often the easiest to carry out, as they can be placed along with the robot. This is very effective when the robot's operation is restricted to a particular area, where each of the markers can be accurately placed to inform the robot. As for the independent system informing the robot, the difficulty lies in the coordination of the appropriate information being passed along, as well as the additional cost in developing and maintaining the other system. Having to manage the other sensors is often the deciding factor when their use is considered. Many real world applications tend to make use of already existing, yet man-made, markers, such as road markings and room number plates.

### 8.1.4 Existing maps

Rather than making use of the sensor signals which portray an immediate measure of the environment, it is also possible to make use of a more complete set of attributes which corresponds to the state of the environment at some time in the past (Zelinsky & Yuta, 1993). These can often be classified as pre-constructed maps, since they contain information about the scene from multiple perspectives and often contain a complete set of attributes that were required at the time of the map

#### 8.1.4 Existing maps

construction. Depending on the purpose of the provided map, the level of accuracy and attributes that are used may differ, as well as the difference in the representation itself.

When converting the provided map to the format required by the robot, the translation of the attributes must be appropriately weighted by the reliability of the map, which often depends on the original intention of the map. As the majority of maps are intended for use by humans, the characteristics that are shown are typically incorrect in proportion and contain higher level concepts which require knowledge of the context, thus do not directly include a lot of information that can be used by the robot. This is typically due to the absolute measures that are used by the robot. The higher level concepts can be copied over to the robot's maps, but they are often not understood by the robot due to the lack of context. When accurately scaled maps are available, such as the blue print of a building, it is important to filter out the irrelevant portions of the map that are not required by the robot.

One of the more commonly seen occurrences of pre-generated maps is the use of older maps generated by the same robot in a different execution. The consistency in the configuration allows for simple integration between the two maps, which allows for a highly confident inclusion of the given set of information. This technique is often used in global localisation algorithms, dynamic object and drifting detection, as the consistency and also the lack of consistency between the maps allow for the differences in the maps to be marked as being an interesting characteristic (Dudek et al., 1997).

## 8.2 Map types

The consideration of how the derived attributes are to be stored and maintained include aspects like the accessibility, memory consumption, manipulation speed, extensibility, as well as the precision or the amount of detail it can contain. The type of map that is used has a direct influence on the internal representation at the low level, thus close coupling is required to the data-structure. As with most applications, it is the context which defines the data-structures to be used, especially due to the flexibility and resource availability of general purpose processors to freely allow the selection of the most appropriate implementation. Although it is possible to make use of dedicated processors, such as graphic processors due to the highly independent nature of map components, the fundamental usage behind the different map types remain the same. With this in mind, various map types are discussed along with the appropriate data-structures that may be used to implement it.

### 8.2.1 Grid

Based on the idea of the metric map, a finite interval coordinate points can be used to define a unit of space for the attributes to be assigned to. By restricting the size of these spaces to a consistent amount, they can be placed in a grid like manner. This configuration allows fast random access to each of the grid cells and can control the memory consumption by modifying the size of the cell. It is quite common to see a Cartesian coordinate based grid map being used, especially when combined with the occupancy of the cell for robot navigation tasks (Borenstein & Koren, 1991).

### 8.2.1 Grid

The memory consumption is often limited in grid maps, as the majority of the implementations use a fixed sized array to represent the map. The regularity of the coordinate points allows inter-neighbouring cell comparisons to be carried out quickly and allows simple storage of attributes, as each of the cells can directly maintain a group of different values. One of the crucial issues with this approach is dealing with increases in the operational range or the number of attributes while maintaining an acceptable memory footprint, as the data-structure doesn't often allow easy expansion, as well as the maintenance of uninteresting portions within the map.

In the event of the robot traversing beyond the initially anticipated range, the map must undergo various strategies to deal with the increase in the map. The three basic strategies that are available are shifting, extending, and scaling. The shifting approach involves the removal of unwanted portions of the map to make way for the new area to be mapped. This allows for the memory footprint to remain consistent and typically only requires a simple change to the underlying data structure, such as the changing of the offset value. This strategy is commonly used when only certain portions of the environment is needed in the map, such as the immediate surroundings, since the discarded portions are reset and used to represent the new area.

When shifting the map, there are two basic strategies to maintain the same grid structure, such that the neighbourhood relationships and the same memory footprint are maintained. The first of the approaches involve copying the cell contents across, such that the coordinate point with respect to the data structure is consistent. With this approach, an offset value from the origin of the map to the grid itself may be necessary for better utilisation of the array. The second approach involves the use an offset value with respect to the grid itself to indicate where the new reference point is for the data inside the array. This approach typically makes use of a circular counting technique, such that the same memory location can be used for multiple indexes without the need for shuffling of the contents.

The copying process for a 2D grid map is typically triggered by the whole row or column being eliminated, thus each of the cells in the perpendicular direction must be shifted by the same amount to fill the now vacant cells. This means the number of copy operations can be a significant amount if the size of the map is large or the is this is frequent. Although the approach is very simple to implement, the second approach is more commonly seen due its effectiveness for all map sizes. Since the shifting that is required is constant for all the remaining cells, this can be converted to an offset value to be used when accessing a particular cell. To re-use the vacant cells, the indexes to the position in memory can be wrapped around, such that the wrapping is invisible from the perspective of the map user. Note that this technique is applicable for cases where the cells being removed is exactly one map width apart to the cells being introduced, as it simply re-uses the memory location instead of allocating more for the new cells.

Although this approach removes the need for the copy process, the introduction of the extra offset means frequent access to the cells in the map will require repetitive division operations to wrap the index values around. This drawback is often neglected and can end up resulting in a reduction in the performance. To reduce this overhead, an extra constraint can be introduced by setting the map size to the power of 2. This then allows the use of a simple bit mask to determine the wrapped offset,

### 8.2.1 Grid

as the positions of the bits stays relative to the array positions. Algorithm 8.1 illustrates the circular indexing of the grid cells where the width of the map is a power of 2, while the height is not. Note the case where the position is negative, which is with respect to the origin of the coordinate axis. This can be ignored if two's complement negative values are used with the bitwise operation, as with the  $w$  index.

```
function GetAttribute(map, i, j):  
    set h = j % map.height  
    if h < 0:  
        set h = h + map.height  
    return map[i & (map.width-1), h]
```

Algorithm 8.1: Circular indexing of grid cells.

The shifting process can potentially consume a significant processing time and can contribute to the corruption of the attributes if the process is not carried out at appropriate moments. In typical scenarios, the robot's pose has a higher level of precision than the one used by the map. This means that the change in the robot's pose will require partial shifts of the array elements, which can be costly and introduce large amounts of artefacts through interpolation. The blurring of the cell attributes can be avoided by a simple technique of buffering the pose changes before the map is updated. The actual buffer size can differ depending on the pace of the robot and the range of the sensors with respect to the size of the map. However, they should be multiples of the cell size, which will avoid the interpolation issues between the cells occurring.

Extending the map involves additional memory being allocated for the new area and merging the two maps together. This approach handles the range issue without compromising the existing information about the environment. However, it can suffer from the bloating of the memory footprint as portions that are unlikely to be used in the future will still remain to be just as accessible. A typical grid map implementation involves a square structure, thus the increase in the map size is proportional to the dimension of the current map, thus can increase significantly if motion in multiple axes are equally present. This inefficiency is often tackled by restricting the motion of the robot rather than manipulating the map.

The last of the approaches, the scaling, involves the compression of the current map and a change in the representative scale for the cell sizes, such that the memory footprint remains the same. This technique is widely used in rendering processes by applying well known algorithms like sub-sampling and various interpolation techniques to remove the uninteresting details of the data without removing much of the interesting or distinctive portions. It is important to observe the decompression time with certain algorithms, as the access time is just as important as the memory footprint. The various attributes within the grid cell and the purpose of the map are often considered to determine the type of compression algorithm, but typically involves a lossy process.

While the cells being updated is important, it is equally important to make sure that the information carried by the cells being discarded are not completely lost and the new cells being introduced consider the availability of information that surrounds the position (Balmelli et al., 1998). For cells being removed, the key contents of the cells should be carried over to another representation, or stored in memory for future



### 8.2.1 Grid

reference, which may be accessed during an off-line processing stage. This process should ideally take place using a number of cells at once to form a higher level perspective of the area for compression, thus require a buffer or a viewing window to analyse the portion being removed.

For the cells being introduced into the map, the contents can be initialised from three potential sources. The first approach simply resets the cells by assuming the portion of the environment has not been explored yet. This allows for a safe but slower approach, as past information from other sources such as other maps are not utilised. This assists in clearly identifying the presence of dynamic objects, as the lack of influence from the old map will help generate an unbiased model of the current surroundings.

The second approach involves the use of another map, where the cells are populated with information generated from the past. This hastens the map generation process, but can potentially introduce errors from misalignment or mislead the process in case changes to the environment occur. There is also the issue with the attributes that are not shared between the maps, which must be initialised by some other means.

The third approach is similar to one that is often seen in video processing algorithm, where the image is extended beyond its boundaries by replicating the bordering textures when attempting to predict the texture outside the current view. Since many man made obstacles, especially those that form structures, have regular shapes, this technique can provide a reasonable model of the newly introduced area, especially if trends like lines can be determined for those that intersect the borders.

To combat the drawbacks of the three approaches above, a hybrid algorithm can be implemented which relies on the reinforcement of cell attributes. By combining the cell information from the other map and the neighbouring cells, similarities and irregularities can be identified to weight the confidence in using the provided attributes. The actual prediction can be derived from one or both the sources, while the confidence weight can shift the attributes from the unexplored state. Algorithm 8.2 below shows an implementation of the update process of the old and new cells using the circular indexing where the average of the two approaches above is used. Only the horizontal shift is shown, as the vertical shift is almost identical in implementation. The variable `another` is used to store a compressed form of the cells being removed.

```
function HorizontalShift(map, delta_x, another):
  if delta_x > 0:
    set to_remove_array[map.height][delta_x]
    for j in 0 to map.height:
      set neighbour = GetAttribute(map, map.x + map.width -
        1, j + map.y)
      for i in 0 to delta_x:
        set to_remove_array[j][i] = GetAttribute(map, i +
          map.x, j + map.y)
        set other = GetAttribute(another, i + map.x +
          map.width, j + map.y)
        SetAttribute(map, i + map.x + map.width, j + map.y,
          (other + neighbour) / 2)
    Merge(to_remove_array, another, map.x, map.y, map.buffer,
      map.height)
```

## 8.2.1 Grid

```

else:
    set delta_x = -delta_x
    set to_remove_array[map.height][delta_x]
    for j in 0 to map.height:
        set neighbour = GetAttribute(map, map.x, j + map.y)
        for i in 0 to delta_x:
            set to_remove_array[j][i] = GetAttribute(map, i +
                map.x + map.width, j + map.y)
            set other = GetAttribute(another, i + map.x -
                map.buffer, j + map.y)
            SetAttribute(map, i + map.x - map.buffer, j +
                map.y, (other + neighbour) / 2)
    Merge(to_remove_array, another, map.x + map.width -
        map.buffer, map.y, map.buffer, map.height)

```

Algorithm 8.2: Removal and initialisation of old and new cells when the map is shifted horizontally.

The components of the grid map can be seen in figure 8.1, while the algorithm for moving the robot can be seen in algorithm 8.3.

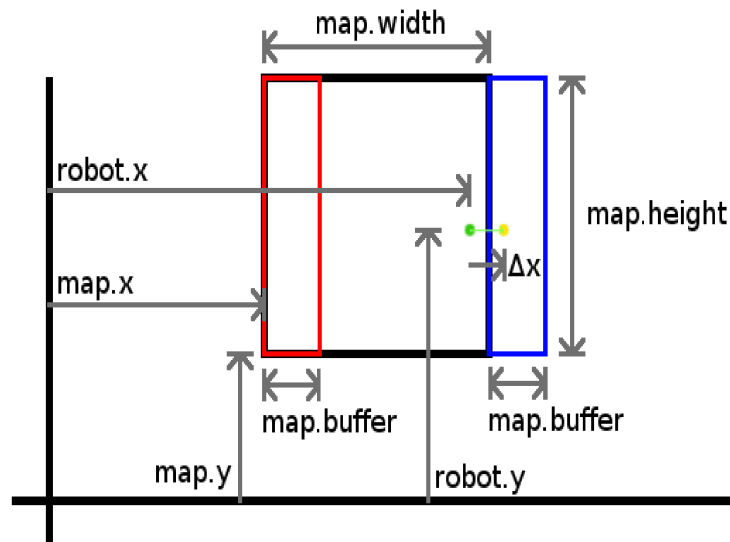


Figure 8.1: Components of the grid map when the map is shifted. The same notation is used in algorithm 8.3, except for  $\Delta x$ , which is called horizontal.

```

function Move(map, robot, horizontal, vertical, another):
    set robot.x = robot.x + horizontal
    set robot.y = robot.y + vertical
    set new_x = robot .x - map.x
    set new_y = robot .y - map.y
    if new_x > map.width:
        set delta_x = map.buffer
    else if new_x < 0:
        set delta_x = -map.buffer
    if delta_x != 0:
        HorizontalShift(map, delta_x, another)
    set map.x = map.x + delta_x
    if new_y > map.height:
        set delta_y = map.buffer

```

### 8.2.1 Grid

```
else if new_y < 0:
    set delta_y = -map.buffer
if delta_y != 0:
    VerticalShift(map, delta_y, another)
set map.y = map.y + delta_y
```

Algorithm 8.3: Pseudo-code for robot motion and monitoring when the map requires shifting.

One of the distinct problems of this approach is that any error in the cells along the border will be carried over to the extended cells to form distinctive streaks that do not correspond to the real world. Although these portions can later be corrected when the sensors scan the regions, these unnatural patterns can cause strange trends to be observed when high level analysis is carried out. Since a detailed analysis on the shape and trends of objects are not carried out at this stage due to performance issues, the reliability of the attributes included in the initialisation must gradually decrease with distance.

To implement this, a corrosion of the weights introduced, such that the scores of the attributes are slowly decreased as they are introduced. It is possible to simply average the neighbours, lowering the weights by a fixed amount or a percentage, as shown in figure 8.2. The first approach allows the objects to spread out, thus promoting true and false positives and can require a reasonable amount of distance from the original position before the attributes are deemed irrelevant. By gradually reducing the weight of the attributes and promoting the uncertainty of the attributes, it provides some level of consistency with the adjacent cells and allows control over how much it should rely on the existing attributes. Since the precision of the occupancy can be unreliable at times, the size of the reference row or column can be increased for a better indication of the state of the space near the map's boundary.

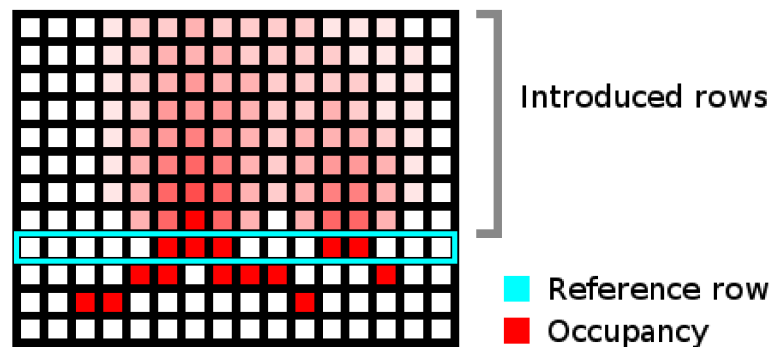


Figure 8.2: Corrosion of attributes in the initialisation of newly introduced cells. The intensity of the cells is the proportional weight used for the attributes.

Note that when shifting is required in both axes simultaneously, the operations for both directions must occur simultaneously to avoid biasing of attributes by sequentially extending from one axis then another.

By allowing the robot's position with respect to the map to change, the map can no longer be modified through a fixed template of sensor interactions, such as fixed positions for shapes and directions of the sensor scans. The variation in the position

### 8.2.1 Grid

introduces aliasing issues, which is also the case when rotations occur. However, by allowing the cell's attributes to rotate with the robot, it causes dispersion and merging as the cell is mapped onto a group of cells after rotation. This also causes problems with the corner areas of the map, as they enter and leave the map. Instead of changing the orientation of the map, the direction of the sensor scans can be modified and the aliasing issue can be dealt with in similar ways to that of the sub-cell sized motions.

Due to the similarity of the grid map and raster graphics, many image processing algorithms can be applied to enhance the performance in the maintenance of the map. This will be discussed in further details in chapter 9.

### 8.2.2 Quad tree

Instead of using uniformly spaced grid cells, it is possible to combine different sized and shaped cells to better utilise each cell. A commonly seen example of this is the quad tree, where the size of the cell is determined by density of the information to be represented (Balmelli et al., 1999). The structured splitting and merging of the cells allow for effective compression of information, but the approach can sometimes be overwhelmed by the modification of the cell arrangements (Yang & Lee, 1994). This can often limit its use during the construction of maps, especially if the cells are merged and divided frequently to increase the effectiveness of the compression. However, this approach is useful in compressing portions of the map that are inactive or may not be of interest, which are problematic in grid maps due to the dense and uniform representation of the surroundings (Chen et al., 1995).

### 8.2.3 Geometric

Based on the group of sensory data, a map can be represented using a collection of geometric shapes (Dudek & MacKenzie, 1993). This representation can consist of constructs such as points, lines, polygons, and so forth, which combines the sensory data by some constraint. This often requires a set of primitives or templates to be used to associate the sensor data, but allows a very compact representation of the environment. The formation of these geometric shapes are also accompanied by set of rules to encourage certain shapes to form over others, such as larger shapes with lower number of vertices over smaller and complicated shapes. Once these geometric representations are formed, manipulation of each construct becomes a simple task. However, the models are often difficult to derive due to the limited sensor precision, variation in the tolerance specified in the template, and the loss of subtle features when approximating to one of the pre-specified shapes.

Although the memory footprint may start off being small, this can potentially grow as more sensor data is introduced and often overlap one other. To overcome this issue, compression mechanisms are required to detect and remove redundant shapes, such as the removal of shapes that overlap. One of the major drawbacks of this approach is the complexity in identifying relationships between the points and structures, as sequence or neighbourhood information is not maintained and would rely on other structure, such as a tree structure to store some form of adjacency information to increase the processing efficiency (Guttman, 1984).

## 8.2.4 Topology

### 8.2.4 Topology

An alternative approach to using a geometric shape based mapping is to identify regions with some form of semantic information (Kuipers, 1978) to describe itself and to connect between them, such as by using graphs, matching against patterns, or apply clustering techniques (Nagatani et al., 1998; Remolina & Kuipers, 2004). Since the number of data points is proportional to the number of scans and is not necessarily the measure of relevance or interestingness, the sensor readings require compression to leave behind only those of interest.

As more and more sensory data is converted to the map, the level of complexity in the map increases and can require additional layers to summarise the map, or at least a sub-portion of the map. The summary, like the template shapes and clusters, can be classified into groups of higher level concepts to form what is called a topological or semantic map (Shatkay & Kaelbling, 1997). They represent the relational and category labels on components within the map, such as corridor and room. The map is typically represented in a graph structure, which allows for high levels of compression and also has commonalities with the spatial perception of humans. Since these high level constructs often do not have a spatial constraint, the use of a graph structure is more suited than the grid representation.

Many of the map types discussed above are often used in parallel for specific tasks. The specialisation allows the individual maps to focus on portraying a particular information, while the combination with other maps allows complimenting information to be derived and stored (Duckett & Saffloti, 2000). To coordinate between them, effective inter-map messages must be devised to allow changes to one map to affect another, thus maintaining the synchronisation (Thrun & Bucken, 1996).

One commonly seen topological map is a connectivity map, which illustrates the neighbourhood of significant points, such as directions between landmarks. Although the information that is represented is quite similar to that of a grid map, the graph based approach allows varying levels of detail between points of interest. This means the details about how they are connected can be stored, but the information with regards to the region in between can be discarded.

Figure 8.3 shows a sample connectivity map, superimposed over a floor plan map to indicate the presence of a path between the nodes. It is important to note that the links between the nodes is not required to represent the immediate neighbours like the grid maps, nor does it indicate a direct line of sight. Instead, the connection simply states that it is possible to get to the other node without directly passing through another node.

One of the most crucial considerations to make when implementing a graph based map is the criteria for creating a node. The analysis of the grid map allows for various attributes to be considered, such as the occupancy of a region and the relationships to the other nodes. This can potentially form the basis for the attributes of the node, as well as deciding whether it is necessary to create the new node within the map. Unlike the previous maps, where the relationships between the grid cells could be derived with ease from the coordinate values, the nodes within the graph must carefully consider the connectivity, such as the necessary motions required by the robot or line of sight, to allow connections between the nodes. However, it is

### 8.2.4 Topology

entirely possible to configure the links such that a different measure is used to link between the nodes, which may include things like decision points in mazes or restrictions in the traversable direction, such as with one way roads.

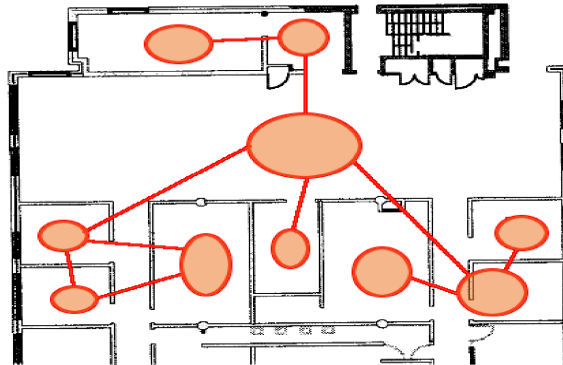


Figure 8.3: Example connectivity map superimposed over a floor plan. The circles in this example represent the nodes containing information about the region.

Once a new node has been placed into the map, the rest of the nodes must be analysed to determine the appropriateness with the current view of the environment. Depending on the level of precision and memory consumption required by the nodes, it may also be feasible to maintain several layers of semantic maps simultaneously to allow analysis of the nodes from multiple perspectives. This may involve the removal of redundant or insignificant nodes, or even a merger between multiple nodes. When merging is required, some information will be lost from interpolation and elimination of certain attributes, thus each node should maintain a record of the number of merging operations it has performed to note the amount of errors that could have been introduced.

The type of map we are most accustomed to is the semantic map, where it portrays a series of very high level concepts with minimal precision and detail of the other attributes. When constructing a semantic map for a mobile robot to use, it is important to note that it is often quite meaningless to the robot itself, as it does not have the contextual understanding of the attributes. It is possible to classify the attributes using templates or cluster them in similar ways to self organising maps, but are typically only useful to human observers thus the formation of topological maps should be regarded as a secondary functionality for compression and for human observers to view later on.

## 8.3 Map layers

The process of creating the map can be categorised into two types, where one involves the maintenance of the immediate surrounding environment, called the local map, and the other involves the whole environment the robot is exposed to, called the global map.

### 8.3.1 Local map

### 8.3.1 Local map

The local map typically extends to the area where the on-board sensors can reach, which means they are frequently accessed and manipulated to reflect the up to date readings of the sensors, thus the overall size of this map is kept small. Since the local map must interpret the sensor information of multiple types, it must be equipped with strategies to convert the various sensor readings into a uniform representation. This allows for simple interaction between the sensors. Figure 8.4 below shows a sample view of the local map, where the red circle is the current location of the robot, blue areas represent the regions that are deemed to be free of obstacles and the green areas are areas where there could be obstacles present.



Figure 8.4: Sample view of the local map.

The yellow represents vacancy, which is carved by the robot's body, the IR sensors and the sonar sensors. The magenta represents occupancy, which is where the range finder scans terminate. The intensity represents the confidence measure.

Since the map must be able to handle the sensor measurements, it must have the capacity to modify the map in all directions the sensors face. This often leads to the centering of the robot within the map, since sensors are placed around the robot in all directions in most configurations. Whenever the robot moves, the map must shift in the opposite direction to maintain the robot close to the center of the map. Note that it does not have to remain perfectly in the center, as long as there is enough room for the sensors measurements to affect the map.

Although the use of a geometric map would allow accurate portrayal of the sensor readings, a grid map allows for an easy base for the various sensor measurements to be merged. Since the length of time the local map stays constant is dependant on the

### 8.3.1 Local map

speed of the robot, using the geometric map could result in large number of shapes being included if the robot does not move around or if the scan rate of the sensors are set very high. Using the grid map allows for fast random access to the sensor data, as well as being able to maintain a constant memory footprint.

Various strategies in using the range finders for forming the local map is discussed in chapter 9, while techniques using visual sensors are discussed in chapters 10 and 11.

### 8.3.2 Global map

Since the scales of the local map remain the same, exploration of a large environment would not allow the robot to maintain any information about how the current view relates to previously visited areas. Although the scales and granularity should depend on the obstacles and the robot's size, it is also important to include the proportionality of the traversal area by the robot. The global map differs from the local map by maintaining a scaled or compressed perspective of the entire environment the robot is interested in. The map allows for a global perspective of the explored environment to allow a large scaled analysis, such as path finding (Low et al., 2002; Masoud & Masoud, 2000), derivation of semantic information, and connectivity.

To control the memory consumption, portions of the map must be discarded when the robot explores a new area outside the bounds of the current map. When determining the compression algorithm, considerations as to what information will be retained and discarded must be made, as well as the potential introduction of ambiguities in the connectivity and alignment due to the change in the scale. Depending on the compression algorithm and the constraints to the robot's operational range, it may be possible to use a lossless algorithm by sacrificing some processing time using a non-lossy compression algorithm or storing the data to a permanent memory for off-line processing.

The majority of the global map generation is done through the information provided by the local map, where periodic updating occurs to fill in the details of the current surroundings by superimposing the local map onto a portion of the global map (Clemente et al., 2007; Williams, 2001). Although the primary function of this will be to fill in the unexplored portions of the global map, it is also possible to use the merging phase as a correlation process to correct the current pose of the robot or to determine any inconsistencies to flag areas of dynamic properties or even errors (Schiele & Crowley, 1994; Weiß et al., 1994). Strategies for implementing a global map by combining the local maps are discussed in detail in chapter 9.

## 8.4 Summary

When generating the map, it is important to consider the characteristics of the available information, such that meaningful information is used and maintained. The selection of the appropriate attribute depends on the specific task of the robot. For a navigational robot, the important attributes include the occupancy and dynamic properties of the obstacles, as well as connectivity between various points the robot



## 8.4 Summary

can manoeuvre. Many other attributes can also be used, but this depends greatly on how useful the extra attributes are to the robot. One strategy that is considered in chapter 11 is the use of sparse landmarks which is maintained in a separate map layer.

Although the accuracy and the level of information that can be provided by artificially crafted markers and external systems can be very desirable, the focus of the current system is to develop a series of algorithms that can operate independently of other systems, thus forming a strong foundation for extension and improvement later on which could include the use of the external systems for support.

While keeping the memory footprint low and reducing the processing requirements for the map is desirable, maintaining multiple layers can assist in simplifying the communication between other systems. As well as the actual information being stored, the various layers need to consider the arrangement of the information, the conversion process between the layers, and the maintenance of the information in terms of how much and how long the information should be stored.

With the idea of specialisation in mind for the map layers, three different layers are implemented to simplify the task at each layer, which are the local, global, and the connectivity maps. The primary focus of the local and global maps are to combine the sensor information quickly, which is achieved through the use of the grid map, while the connectivity map is used to store higher level concepts to be used for complex decision making and to allow a more tangible representation to the observer. This map also maintains some of the vital information about the structure of the environment that may be lost when the scale for the global map becomes too large.

## Chapter 9 – Carving using range finders

Due to the simple process that is required for a uniform representation for the sensory information to bind to, a 2D Cartesian coordinate based grid map is used to represent the local map of the robot. The highly dense and memory consuming nature of this type of map allows an accurate portrayal of the environment, while providing quickly accessible information between different locations within the map. This characteristic allows for fast modelling of the sensor readings while maintaining a consistent memory footprint.

As the current task for the robot is to model the structure of the environment, the emphasis is placed on the occupancy of areas within the map where the robot can observe. Although it is ideal to construct a 3D representation of the environment, many of the sensors on board are not capable of modelling this due to the lack of elevation control on the robot and also on the viewing angle (Katz et al., 2005). For this reason, the map is constrained to the 2D representation of obstacles, which is still capable of high level tasks, such as path finding and segmentation of the scene to identify sub-components (Shi & Malki, 2000).

One of the key strategies used in this particular implementation of map formation is the idea of a multi-layered representation of the environment. The local map is used as the initial interface for the raw sensor measurements, while the global map is derived through the layered superimposition of the local maps. The range finding sensors, which are the sonar and IR sensors, are considered for the construction of the local map, which can identify where the obstacles lie using a technique known as carving (Burgard et al., 1999). The basics of this technique involve removing the occupancy of regions that overlap with sensor scans.

### 9.1 Occupancy map

Using the current array of sensors, it is possible to obtain directly or to derive a wide variety of attributes about the environment, such as the acoustic reflectivity and texture information. However, many of these are not necessary for the majority of map construction phase, as the map is mainly interested in the physical occupancy of regions. The implementation of occupancy maps can include simple binary occupancy levels, a counter based accumulator, or a probabilistic model, which can be similar to a normalised implementation of the accumulation approach (Martin & Moravec, 1996; Wijk & Christensen, 2000).

#### 9.1.1 Sensors and attributes

The complete construction of the occupancy map involves the use of the majority of the sensors on the robot, including the IR array, two sonar sensor modules, a webcam mounted on top of a servo motor, and another webcam as part of the omnidirectional camera module at the top of the robot. The sensors being used can be

### 9.1.1 Sensors and attributes

categorised into two distinct groups. The IR and sonar sensors are grouped as the range finders, while the webcams are categorised as directional sensors, as they measure the incoming light's intensity at various angles. Since the two categories behave significantly differently to each other, they will be treated separately and any information that is derived will be merged at a later time. The considerations for the directional sensors are discussed in further detail in chapters 10 and 11.

One of the effective characteristics of the range finders is the ability to determine the distance to an observed object. This indirectly allows the region before the object to be marked as vacant, as the range finder requires a direct line of sight between the sensor and the object. Note that the signals actually determine the distance to the surface which could not be penetrated by the sensor signal, thus can sometimes be misleading. Although the obstacles indicate the areas that cannot be traversed by the robot, there is also another attribute related to how hazardous the area is. This being an important attribute for the safe operation of the robot. The use of this attribute will be discussed in chapter 11.

Depending on the modality used by the sensors, other attributes, such as the reflective properties and surface orientation (Araujo & Grupen, 1998; Lacroix & Dudek, 1997) can sometimes be determined indirectly, which can also be used to assist the formation of the scene structure. Many of these extra attributes will not be included in the discussion, as they require specific sensor arrangements or multiple scanning of the environment to observe the inconsistencies in the sensor measurements.

Other attributes that are maintained within the cells include the frequency count of how many times the cell has been modified by the sensor scans, the pose of the sensor which affected the cell, the time stamp of the last access to the cell, and finally the surface orientation, which will be useful when constructing object surfaces from the combination of the cells. These attributes are put in place to allow for a more robust carving algorithm, such that it minimises the effects of erroneous sensor readings and repeated sensor readings from the same perspective.

Using the orientation of the robots when the scans are made, the surface orientation can be measured by observing the incoming angles of the sensor beams and noting the range in which the sensor signal was successfully reflected. This is achieved by maintaining an average of the perpendicular direction to the sensor using the frequency counter. As the value represents an angle, this value can cycle around and also does not have a specific initial value.

When observing the behaviour of the sensors, it was noted that fluctuation in the sensor readings fell within two distinct categories. One was where the value fluctuated by a small amount, possibly due to sensor and ambient noise, while the other was when an erroneous reading was made. The distinction could be made quite easily by using a median filter for the IR sensor and a confirmation check for the sonar sensor.

The implementation of the median filter involved a simple buffer that was maintained to filter out the outlier value amongst the two adjacent measurements. This was achieved using a cyclic buffer, which encouraged smooth transitions in the measurements. This approach meant that the IR sensors would misalign when there was a sudden change in the distance to the object, but the high rate of scans

### 9.1.1 Sensors and attributes

compared to the speed of the robot meant the error would stay quite small.

As for the sonar sensors, although the arrangement of the two receivers can potentially allow for narrowing of the obstacle's location through triangulation, the level of precision that can be achieved does not allow for a reliable distance measure. Instead, the two measurements are used to identify any inconsistencies in the measurement to flag the cases where the orientation of the object's surface did not allow for one of the sensors to receive the signal or when an obstacle was not in line with one of the sensor's viewing area. To implement this, the two measurements had to be within 700 units of each other, which equates to approximately just under 10 cm.

### 9.1.2 Carving

The carving process can be implemented using several different approaches. A commonly used technique is to only note the presence of obstacles by initially assuming that all areas are occupied and removing those regions that are traversed by the sensor signals. The opposite approach of assuming all regions are unoccupied is also commonly seen, where the locations of the objects detected by the sensors are marked. Other techniques include the formation of two maps, where one is used for the vacancy and the other for occupancy, or the combination of the two together as they are mutually exclusive. Another measure that is commonly used in conjunction is an uncertainty measure, which is sometimes used together with the occupancy and vacancy, such as the difference between the two values. Using a single range, it is possible to represent the three together where a low value represents vacancy, a high value represents occupancy, and the mid point represents uncertainty.

When the sensors are used to carve out an area, the shape of the region being affected will depend on the sensor characteristics. In simplified implementations, the effective area is confined to a straight line extended in a perpendicular direction from the sensor, such as the case with laser range finders (Wallner et al., 1997). This leads to a very fast processing time, as it simply requires a 2D ray tracing to mark the cells involved. This approach assumes precise information of where the sensor has interacted with, thus does not suffer from ambiguity issues in terms of where the sensor signal has reflected from. This behaviour often does not portray the actual behaviour of the sensor signal, as many of the devices exhibits a dispersion from both the sensor and the reflected surface. Although it is possible to treat the shape as a straight line, appropriate algorithms must be implemented to correct any false flagging of cells as obstacles and better utilise the actual characteristics of the sensor signals (Pfister et al., 2002).

Using a more realistic model of the sensor signal shape, a sector like shape can be considered. The angle of the spread is not always consistent, but can typically be approximated during the calibration phase for each of the devices. Sometimes, different weighting functions are applied to reflect a more realistic shape, but the weights are generally applied after the sector has been established to isolate the cells that are affected by the scan. Instead of using a sector, it is possible to use a combination of other primitive shapes to represent the shape, namely a triangle and a circle with portions of overlap. The circle, or the convex hull of the arc, can sometimes be decomposed into small triangles, which can potentially increase the

### 9.1.2 Carving

efficiency due to a simpler mapping of the shape's boundary onto the grid map. The geometric shapes can be decomposed further into the lines defining the bounding area. With the outer bounds established, the sector can be filled with simple and efficient algorithms (Henrich, 1993). Figure 9.1 illustrates two approaches to carving, where the left shows just the potential position of the obstacles, while the right shows the vacancy, occupancy, as well as the uncertainty based on how green each cell is.

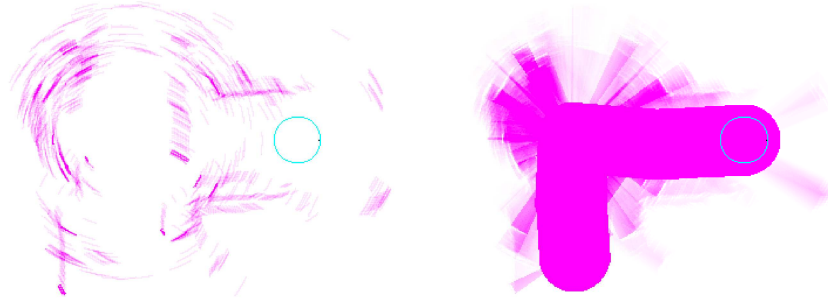


Figure 9.1: Occupancy map carving.

The left image shows the potential location of obstacles from the range finder scans, while the right image shows the vacancy. The small ring represents the robot's body.

The availability of the two types of sensors allow for a wider range of information to be captured, in particular the confirmation of the sensor readings, as the sensors behave differently against obstacle surfaces and have different operational ranges. This can be exploited to encourage the use of both the IR and sonar devices to contribute to the carving, rather than simply using the one with more reliability or consistency. With the current sensor configuration, the sonar sensor beam overlaps with the IR beam at the front end of the robot between the ranges of approximately 400 to 800mm, as shown in figure 9.2. To maintain the difference between the two types of sensors, the attribute for storing the time of last access can be maintained individually for each of the two types of sensors. This means the sequence of processing the two sensor modules will not bias the measurements, as well as allowing confirmations to be carried out if the time of last access is near identical.

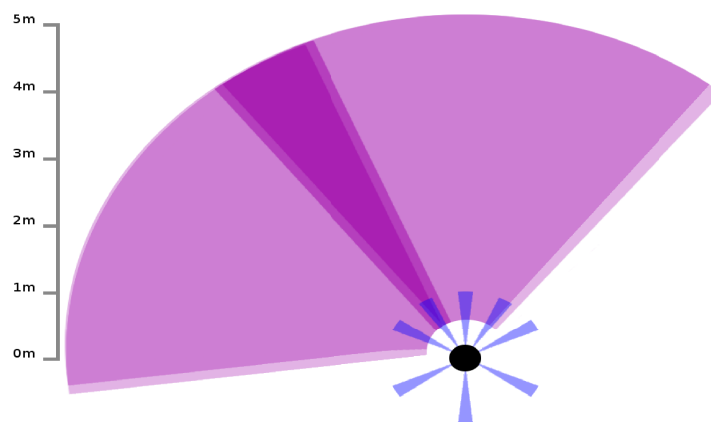


Figure 9.2: Range and effective area of the sonar and IR sensors. The large arcs represent the sonar scan area, while the shorter arc regions represent the IR sensor scan area.

### 9.1.2 Carving

Since the robot will approach many of the obstacles from the front, the sonar sensors should attempt to face forward and the robot should not need to move closer than little under 800 mm to an obstacle to get a distance measurement. This, of course, depends on the reliability of the individual sensors at various orientation and ranges, but can be kept in mind as a simple rule to fall back on for the navigation algorithm.

## 9.2 Local map

As the local map reflects the current surroundings of the robot, the carving process must consider the change in the pose of the sensors. The behaviour of the sensors can differ significantly with both sensor and detected object's orientation, thus should influence the appropriateness and weighting when the sensor scans are translated onto the map.

### 9.2.1 Sensor pose

By measuring the difference in the time to the previous sensor scan, it is possible to modify the weight of the current scan such that rapid scans of the same area does not influence the map as significantly as after waiting for some time to pass. This rule encourages the robot to move around and explore large areas before returning to analyse an already visited region. It also avoids having to use a slow scan frequency which requires normalisations to avoid local maxima from forming if the motion is not consistent. Similarly, using the change in the position of the robot for the weight also encourages the robot to move around and observe the environment from multiple perspectives.

Although the two attributes above sound reasonable, the main contributor to the ideal operation of the sensors is the correct sensor to surface orientation (Grabowski et al., 2003). This is primarily due to the sensor signals being unable to reflect back to the device if the orientation is not within a particular range, which can cause incorrect distance measures to be made. By storing the orientation of the sensors modifying the cells, it encourages the robot to observe the obstacle from multiple perspectives (Feder et al., 1999). This will eliminate some of the problematic issues with the approaches proposed earlier where the motion is not consistent, such as when the robot simply moves back and forth against the object.

To illustrate the effects of the pose based weighting, three different approaches to objects are analysed involving the robot moving directly towards the obstacle, moving parallel to the obstacle's surface, and rotating the robot around the center to allow multiple orientations. The three weight functions that are used include a simple counter, a difference in the positions, and a difference in the orientations based approaches. The interval between the scans is set as consistent to simulate the current behaviour of the sensors, thus the difference in time based approach is not illustrated. However, the end effect is similar to the distance based approach, with the exception of the rotation, which is similar in behaviour to the orientation based approach. The simulation is carried out for 10 scans of encountering a flat surface, a small object, and an angled surface.

### 9.2.1 Sensor pose

When the robot moves towards the object's surface, the sensor reading adjusts inversely to flag the overlapping region. Using a simple counter, the arc region is marked as containing the object, which can change in length, but still marks multiple cells as being consistently and highly likely to contain an object. The same occurs for a small object, thus the two objects are indistinguishable and requires an alternate measure to indicate the true location of the object. The distance measure resulted in a similar result as above, as the motion encouraged the accumulation of the scores to flag several cells as containing the object. Using the orientation weighted approach; the cells are almost unmodified by the subsequent scans, as the orientation does not change. This results in very little changes to the map thus does not allow any confident decisions to be made and awaits for the robot to approach the obstacle from a different angle. When facing an angled surface, the arced portions often do not overlap with each other, thus the only algorithm that is effective is the negation of the occupancy when the portions of the arc overlaps with a non-arc region. Figure 9.3 below illustrates the three approaches, where the score of the counter based approach has been scaled down to a range of 0 to 1. The red regions indicate the possible locations of the object, while the blue regions indicate the regions of vacancy.

For the case where the robot moves parallel to the object's surface, the cells that are encountered the most is in fact those which lie slightly in front of the surface. This behaviour occurs from the fact that the actual surface is reached by a small tip portion of the arc, which results in a jagged surface being modelled. Although this can later be analysed at a higher level to be converted to a flat surface, it can contribute to misalignments of the surface if the robot is moving at different speeds. Depending on whether the vacancy is or is not used in conjunction, some of the false positive objects can be eliminated as they occupy a space that is marked as vacant by different scans. Note that this can create large gaps between the local maxima depending on the interval of the scans.

As figure 9.4 shows, the detection of small objects produces a highly desirable result for two of the approaches, where the score at the obstacle is higher than the rest. Depending on the number and the proximity of the scans, the difference in the counter values will change, thus a filtering process is often required to identify the local maxima or normalised using the total number of scans.

Although the angle from the sensor to the cell changes between the scans, this is undetectable by the sensor and the sensor's orientation remains in the same direction. This causes the orientation based weight approach to map the same shape, but with low level of confidence, thus preventing false positive objects from being formed. When the approach is applied to a small obstacle, the technique is unable to capitalise on the repeated access of the same cell since the orientation remains the same. Once again, this approach requires re-scanning of the area at a later stage when the orientation of the sensors are changed before any confident measures can be established. Observing the angular surface resulted in the same behaviour as the previous robot motion, but with the local maxima being wider apart.

The last of the scenario, which consists of rotating the robot around the center, resulted in similar behaviour to the previous case for the flat and small objects, except with the two weighted approaches being swapped around. The behaviour for the angled surface showed that the orientation based approach formed the jagged

### 9.2.1 Sensor pose

surface behaviour, much like the simple counter based approach. Figure 9.5 illustrates the behaviour for the last scenario.

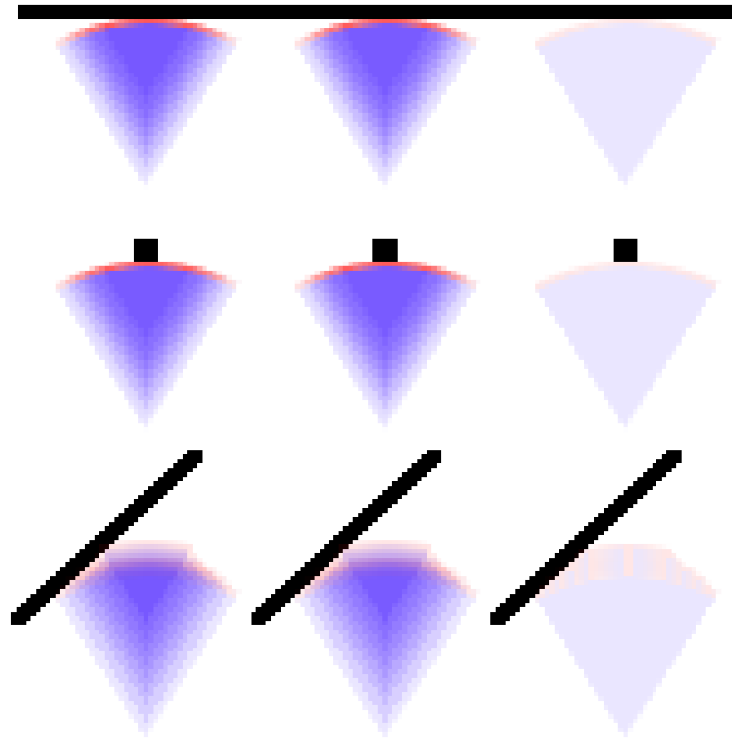


Figure 9.3: Motion towards the object.

The rows represent the different type of obstacle encountered, while the columns represent the three score accumulation strategies. The left being the simple counter, the middle being the position based, and the right being the orientation based strategies.

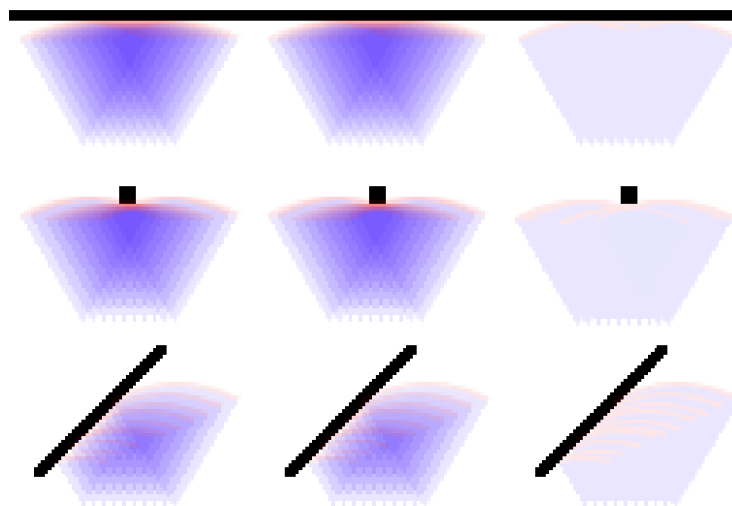


Figure 9.4: Parallel motion to the object.

The rows represent the different type of obstacle encountered, while the columns represent the three score accumulation strategies. The left being the simple counter, the middle being the position based, and the right being the orientation based strategies.



### 9.2.1 Sensor pose

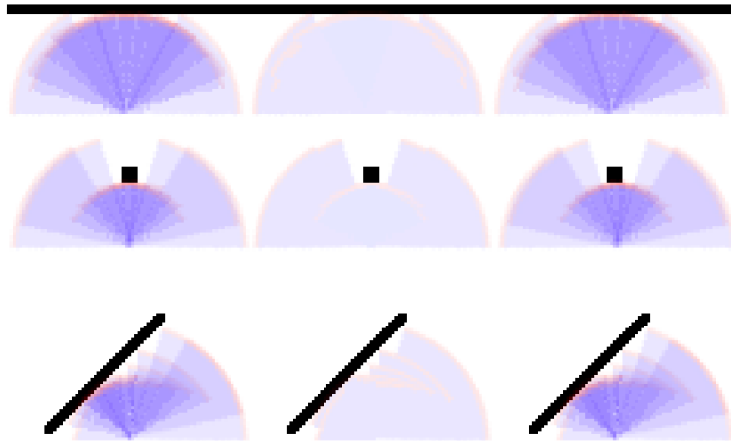


Figure 9.5: Rotation around the center.

The rows represent the different type of obstacle encountered, while the columns represent the three score accumulation strategies. The left being the simple counter, the middle being the position based, and the right being the orientation based strategies.

In all of the approaches above, the cases where the object is detected slightly before the surface can be corrected by increasing the scan frequency. This means that the only problematic cases are the counter and position based approaches when heading straight towards a flat or small object, as this incorrectly marks multiple cells as being occupied. To avoid this from occurring, the robot's motion should encourage rotation, such that the sensor scans are at varying angles to avoid the undesired overlaps.

Using the orientation based approach does not allow the quick detection of objects, as the robot would not be rotating while traversing to various locations, but at the same time, does not form false positives. This strategy can be utilised for the sonar sensor by constantly rotating the sensor using the servo motor. It is also possible to avoid using the sensors located at the front and back of the robot, thus eliminating the first scenario from occurring. This will allow the position based approach to operate without the problematic case. An important issue to note is that the real motion of the robot is never perfectly translational or rotational, thus the cases with little effect on occupancy detection will generally find an obstacle at or near the local maxima.

### 9.2.2 Weights

In many experimental trials, robots manoeuvre around in a set path and focus on covering more distance than repeating scans in the same area to improve the accuracy and attempting to disambiguate faulty measurements. This means many of the scans are spaced apart while measurements done around the turning points sometimes contain more scans. The frequency of the scans thus reflects the speed of the robot, rather than the complexity or importance of the region. Although the increased turning points may indirectly indicate the complexity of the surroundings, this would only be applicable if the robot's motions were based on the presence of obstacles. The simple rule can be used to normalise the points to discard or scale

## 9.2.2 Weights

redundant measurements so the density information can be used to distinguish between an actual object and an erroneous reading. The approach is typically used in conjunction with the vacancy counter to remove the false positive values first, such that the surfaces that are formed do not include rough arcs.

Another commonly used approach is the detection of local maxima by simply observing the scores between the neighbours. One of the issues with this approach is the difficulty in identifying surfaces, as the number of scans dictates how prominent the object is amongst false positives. This technique is used when specific points of interest is desired, such as corners, which can be used in a post-processing phase to construct a surface by connecting straight lines between them. This, of course, places certain conditions on the types of surfaces it can observe.

When the scan is carried out in quick succession, the sensors are not given enough time to move or change its orientation in case the sensor signal is incorrectly reflected. This is also true for the time interval, which can be too small to notice any dynamic behaviour of the objects. By allowing the interval to become large enough, the sensor reading can be given a significantly more weight to emphasise the importance of the particular scan. Rather than simply using a threshold value to specify when the next scan can occur, a weight can be determined to scale the value depending on the length of the interval. This is because the setting of a threshold can cause an important scan to be missed, as the limitation in the reflecting angles may not allow the surface to respond to many orientations. Since the maximum time or position difference is unbound, the difficulty lies in finding a reasonable range to spread the weight. This can be easily controlled for the orientation based approach due to the cyclic nature by assuming that the cell size is small enough for one surface.

Using the attributes introduced earlier, the weighting function can allow the scores to be accumulated more effectively. However, it is important to note the range, as well as the non uniform number of scans that are carried out. To counteract this, the number of scans that have accessed the cell can be maintained, the vacancy and occupancy can be stored as separate measures, or the weight used to apply the change in the occupancy can be proportioned using the current value.

The use of the counter allows simple calculation of the hit rate for a cell, which can be used to normalise the scores. By maintaining a separate measure for the vacancy and occupancy, the two can be superimposed to note any inconsistencies, as well as being able to maintain the number of accesses separately without it cancelling each other out. The last approach can be implemented by observing the current value, then weighting the change as a proportion of the remainder. This will contain the value within a known range at all times, but can lead to rapid fluctuations if only one value is maintained for the occupancy and vacancy. This issue can lead to significant biasing of the last scan, as any change in whether it is occupied or vacant will modify the score dramatically if the observation contradicts the previous observation.

With this in mind, the current implementation uses the orientation based weight by noting the difference in the angles from the last access and scaling the range between 0 and 1. Figure 9.6 below illustrates the weight function based on the orientation.

## 9.2.2 Weights

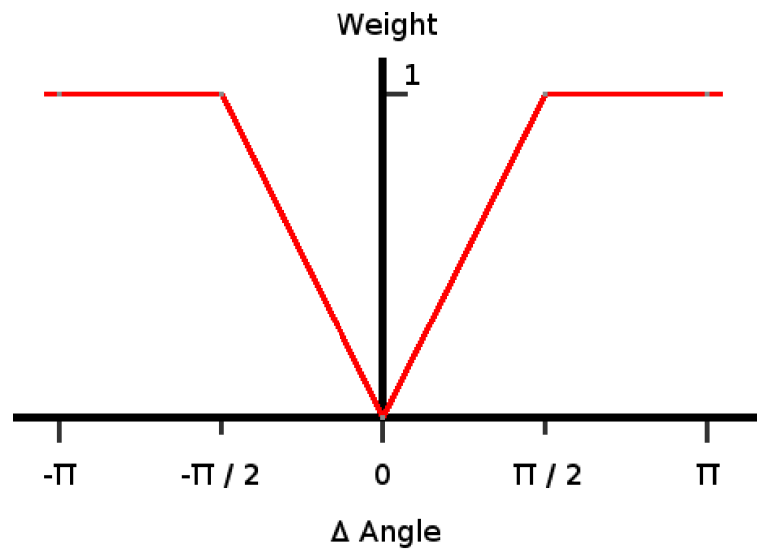


Figure 9.6: Occupancy and vacancy weight modification. The red line illustrates the weights used for a given difference between the previous and current sensor scan.

To distinguish the difference in the reliability between the two types of range finders, another weight is introduced to scale the effect of the sensor readings based on the typical fluctuations in the measurements. Although the fluctuations depend on the distance to the object, as well as the surface types, the reliability can be generalised in a rough manner by observing the consistency in various environments. In the current implementation, the sonar sensors are given a higher weighting due to the consistency in operation with a wider range of surface orientations. Since the majority of the fluctuation occurs from the surface orientation, which cannot be determined, the derivation of the most appropriate weight is difficult to achieve. Instead, an arbitrary value of 0.15 is currently used for the IR sensor scans and 1 for the sonar sensors. Using the low scaling coefficient reduces the rate of change in the scores, thus preventing large fluctuations in the occupancy score if the cell is mistakenly observed.

It is also possible to apply weights to the cells within the sector to account for variations in the sensor behaviour, such as using the distance or the angle away from the perpendicular direction of the sensors. However, this approach is not pursued here as the model depends greatly on the reflective properties of the obstacles, as well as the sensitivity of the sensors. One simple addition which could be considered in future implementation is the use of the standard deviation values or the difference in the distance to the obstacle due to the spherical surface in the distance measures to create a region of potential termination of the sensor scan instead of the constant shape. This may require a more controlled calibration process to determine the appropriate weight functions to be applied to result in something resembling figure 9.7, where the brighter portion represents a higher weighting. Note that the reverse will be applicable for the vacancy, where the black regions will be more likely to be vacant than the white regions.

## 9.2.2 Weights

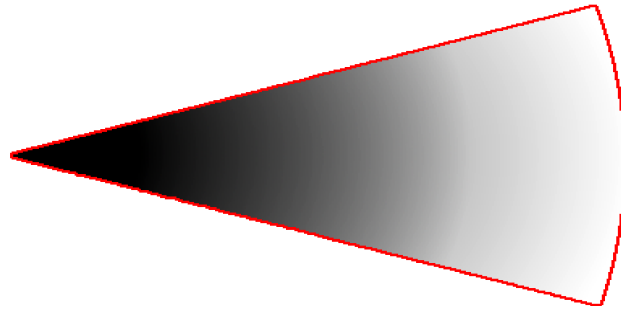


Figure 9.7: Weight function based on position within the sector. The grey scale indicates the weights used, where white represents high and black represents low values.

When observing the behaviour of the sensors on various surfaces, it was noted that although a surface may be correctly detected when observed from a particular orientation, if the sensor's orientation is changed so it is too close to being parallel to the surface, the scans will miss the surface and return an erroneous measurement. By maintaining the occupancy and vacancy as different values, it is possible to detect this occurrence by observing that both values are high.

It is important to note that the false positives will also be flagged as being a real object due to the potentially large number of overlaps. To distinguish this, the surface orientation can be used or a higher level analysis can be carried out based on the connectivity of vacant regions and the proximity of clusters of occupied cells.

Since the surface orientation is gradually derived, this may not be a reliable source to make use of. Similarly, the change in the distance measurements can be attributed to many different events, thus the unexpected bypassing of obstacles cannot be the solely attributed to incorrect orientation of the sensors.

A simple implementation to counter this is to note that the majority of objects are continuous, which suggests that the change in the distance measure should remain small if the same structure is being observed. This technique is used for the sonar sensors, which requires the two readings to be similar to each other. As for the IR sensors, the detection of an erroneous reading from sudden increase in the distance measure remains a difficult task. Rather than reducing the overall weight of the IR sensor measurement based on the distance measure, the operating range of the devices is kept small to avoid the possibility of secondary reflections that causes objects to appear behind another. Note that the median filter is able to remove many of the noisy readings which are often caused by the sensor and not the obstacles.

By reducing the operating range, it also limits the positions of the spurious objects which can often be placed inside real objects. Since the sensor scans are carried out frequently, the boundaries formed by the real objects can be used to eliminate the faulty objects using a filling algorithm to identify the actual objects.

## 9.3 Carving

The carving process itself must occur very quickly and efficiently, especially when the vacancy must be modified. This is due to the large area the sensor scan can

### 9.3 Carving

cover, and the computational cost involved in the translation of the geometric shape to the discrete grid cells. Since the boundaries of the sector can be determined from the distance measure and the viewing angle of the sensor, it is possible to determine if the cell's coordinate point lies within the sector, given the position of the sensor. This reverse lookup process can be sped up using a simpler boundary, such as a bounding rectangle, to reduce the search space for checking the inclusion of the cell. Algorithm 9.1 can be employed for the reverse lookup process:

```
function ReverseLookupSector(map, sensor):
  for j in 0 to map.height:
    set relative_y = j - sensor.y
    for i in 0 to map.width:
      set relative_x = i - sensor.x
      set distance =  $\sqrt{\text{relative\_x}^2 + \text{relative\_y}^2}$ 
      if sensor.distance >= distance:
        set angle = 2 * ( $\tan^{-1}(\text{relative\_y} / \text{relative\_x}) -$ 
          sensor.orientation)
        if sensor.view_angle <= angle:
          if sensor.distance == distance:
            ApplyOccupancy(map[i, j])
          else
            ApplyVacancy(map[i, j])
```

Algorithm 9.1: Applying occupancy or vacancy using reverse lookup.

The idea of converting the sector into a series of geometric shape can be employed, which can allow for a faster traversal of the shape's boundaries. By first identifying the boundaries, it is possible to apply a filling algorithm with a linear scan. In a similar fashion to the above algorithm, the two lines and the arc can be traced, while maintaining the minimum and maximum values for each row, or column, then simply iterating through the stored ranges to apply the vacancy. Algorithm 9.2 illustrates this process. Note that for simplicity, the algorithm ignores some of the boundary conditions and only considers the case where the sector is in the first octant. This eliminates the issue of repeated values along the perpendicular direction to the one being traced due to aliasing, which will be discussed in more detail later.

```
function BoundarySector(map, sensor):
  set low.x = sensor.distance * cos(sensor.orientation -
    sensor.view_angle / 2)
  set low.y = sensor.distance * sin(sensor.orientation -
    sensor.view_angle / 2)
  set low.slope = low.y / low.x
  set low.const = sensor.y - low.slope * sensor.x
  set high.x = sensor.distance * cos(sensor.orientation +
    sensor.view_angle / 2)
  set high.y = sensor.distance * sin(sensor.orientation +
    sensor.view_angle / 2)
  set high.slope = high.y / high.x
  set high.const = sensor.y - high.slope * sensor.x
  set min = sensor.x
  set max = low.x
  set min_array[max - min] = { map.height... }
  set max_array[max - min] = { 0... }
  for i in 0 to low.x:
    set j = sensor.y + low.slope * i
```

### 9.3 Carving

```
    if min_array[i] > j:
        min_array[i] = j
for i in 0 to high.x:
    set j = sensor.y + high.slope * i
    if max_array[i] < j:
        max_array[i] = j
set r_squared = sensor.distance2
for j in high.y to low.y:
    set i =  $\sqrt{r\_squared - (j - sensor.y)^2}$ 
    ApplyOccupancy(map[i + sensor.x, j])
    if max_array[i-1] < j:
        max_array[i-1] = j
for i in min to max:
    for j in min_array[i-min] to max_array[i-min]:
        ApplyVacancy(map[i, j])
```

Algorithm 9.2: Applying occupancy or vacancy using boundary identification and area filling.

Both algorithms above allow the appropriate grid cells to be found, but it does not deal with partial coverage from aliasing errors. At the same time, it is possible to improve the line and arc tracing algorithm to eliminate some of the redundant computation involved.

#### 9.3.1 Anti-aliasing

The problem of aliasing is often ignored by many mapping tasks or reduced by using a smaller granularity for the cell sizes based on the level of precision that can be achieved by the sensor scans. However, efficient anti-aliasing approaches can be implemented to derive the proportional coverage of the cells using various sampling techniques (Schilling, 1991). A sample set of anti-aliasing algorithms are illustrated in figure 9.8, where the blue border represents the areas where the selection can occur, while the red dots indicate the super-sampled cell. Certain algorithms, such as randomised sampling algorithms, are more suited for irregular or unknown shapes occupying the cell. However, for regular or known shapes, the super-sampling algorithm can be controlled to exploit the expected coverage of the cell.

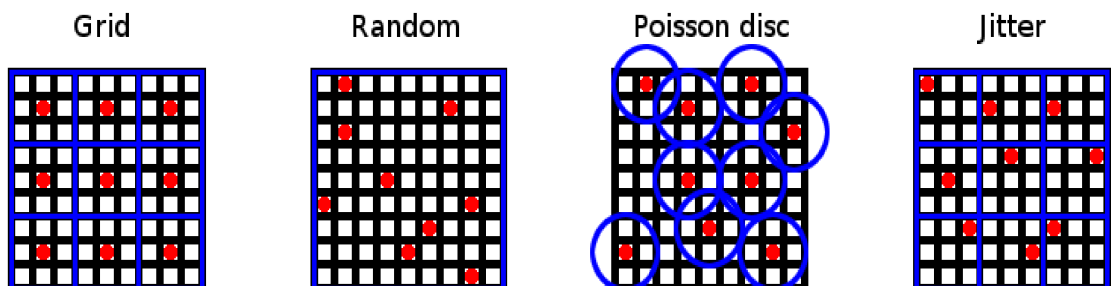


Figure 9.8: Sample super-sampling algorithms for anti-aliasing. The red circles represent the cells selected, while the blue square or circle represent the region in which each of the selected cells occupy. The random case contains overlapping blue squares, thus allowing multiple selected cells to be within one blue region.

### 9.3.1 Anti-aliasing

It is important to try and make use of high precision data when they are available before applying any anti-aliasing algorithms. This includes values such as the robot's pose over the apparent position in the map, or the sensor values over the truncated distance values, as the aliasing can quickly lead to the degradation of the map's quality.

#### 9.3.1.1 Line

Given the line equations, it is quite simple to implement a line tracing algorithm, which will efficiently trace the cells being intersected since the slope of the line stays consistent. Although Bresenham's algorithms are efficient in tracing the line, the aliasing caused by the cell size is undesirable (Bresenham, 1965). Wu's algorithm, on the other hand, allows for sub-pixel precision line drawing, but does not directly translate to the filling that is required (Wu, 1991). Figure 9.9 illustrates the components of the Bresenham's line drawing algorithm. Note the importance of splitting the problem into mirrored and flipped version of the first octant. This assumption allows the algorithm to be simplified, as it restricts the horizontal and vertical motion to fall into two distinct cases.

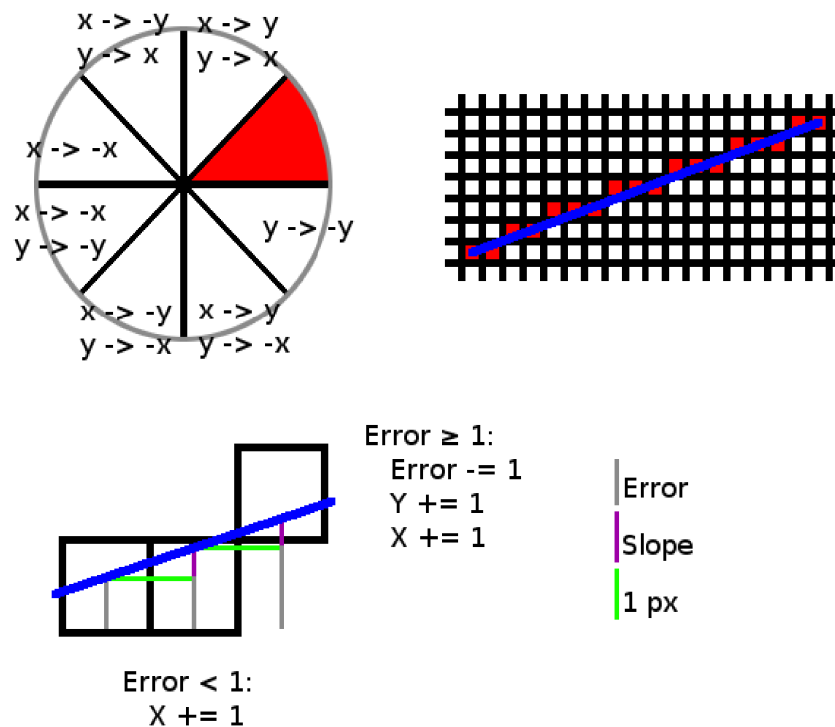


Figure 9.9: Components of Bresenham's line drawing algorithm. Top left shows the translation of the coordinates to bring it back to the first octant, top right shows an example line being drawn using Bresenham's algorithm, and the bottom image shows the two cases where one is a horizontal transition and the other being a diagonal transition.

Based on the conditions used in Bresenham's algorithm, it is possible to devise an anti-aliasing algorithm using the accumulated error values, or the intercept (Pitteway

### 9.3.1.1 Line

& Watkinson, 1980). The two kinds of shapes intersecting the cell can be determined to be a triangle and a rectangle. The triangle represents the upper region of the coverage, while the rectangle extends from the bottom of the triangle to the bottom of the cell. Since the slope remains consistent throughout the line, the intersects defining the points between the triangle and the rectangle can easily be traced. The special case, where there is an overflow in the intersect offset to the one above, results in a portion of the triangle being registered for the upper cell, while the area of the bottom cell must be reduced to account for the missing tip of the triangle.

The characteristics of the two basic shapes are easily determined through the slope of the line and the intersecting point. The area of the triangle always remains the same, while the area of the rectangle is simply the intersection offset, as it spans across the whole cell. Handling the special case, where the top of the triangle requires trimming, involves slightly more work as the vertical intersecting point now needs to be evaluated. Determining when an overflow occurs is a simple task of checking for the difference in the rounded down intersect points at the left and right hand sides of the cell. Using the difference between the intercept at the right hand side and the rounded down value, the area of the triangle can be derived by determining the distance from the vertical intersect point and combining it with the slope of the line. Since the algorithm assumes operation in the first octant, it is important to inverse the values depending on which octant it is actually applied to and which side is within the sector. Figure 9.10 illustrates the various components used in the area based anti-aliasing algorithm, while algorithm 9.3 describes the sequence of operations.

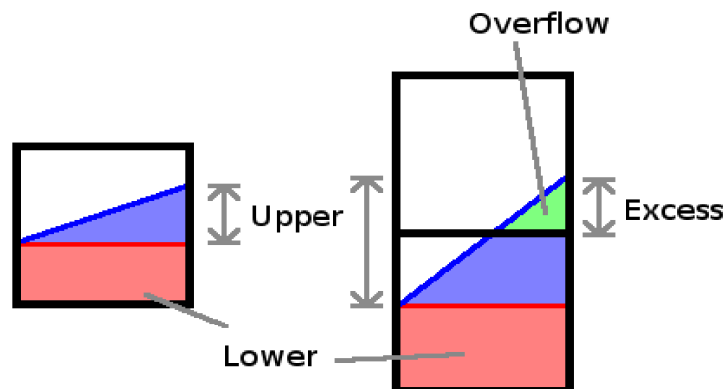


Figure 9.10: Components of the area based anti-aliasing algorithm. Left image shows the simple case where the line is contained within one cell, while the image on the right shows the case where the line intersects a vertical cell boundary.

```
function TraceUpperLine(map, xi, yi, xf, yf):
    set x = ⌊ xi ⌋
    set y = ⌊ yi ⌋
    set slope = (yf - yi) / (xf - xi)
    set triangle = slope / 2
    set endi = 1 + x - xi
    set lower = yi - y
    set upper = slope * endi
    set area = endi * (upper / 2 + lower)
    set lower = upper + lower
```



### 9.3.1.1 Line

```
set excess = lower - 1
if (excess > 0):
    set overflow = excess ^ 2 / slope / 2
    carve(map, x, y, area - overflow)
    area = overflow
    set y = y + 1
    set lower = excess
carve(map, x, y, area)
set x = x + 1
for i in « xf » - x to 0:
    set area = lower + triangle
    set lower = lower + slope
    set excess = lower - 1
    if (excess > 0):
        set overflow = excess ^ 2 / slope / 2
        carve(map, x, y, area - overflow)
        set area = overflow
        set y = y + 1
        set lower = excess
    carve(map, x, y, area)
    set x = x + 1
set x = « xf »
set y = « yf »
set endi = xf - x
set upper = endi * slope
set endj = yf - « yf »
set lower = endj - upper
set area = endi * (upper / 2 + lower)
if lower < 0:
    set overflow = endj ^ 2 slope / 2
    carve(map, x, y, overflow)
    set area = 1 + area - overflow
    set y = y - 1
carve(map, x, y, area)
```

Algorithm 9.3: Coverage area with line tracing.

Since the arc is placed at the end of the triangle, it is not necessary to complete a triangle shape before traversing the arc to form a bound for the sector. When executing the algorithm for the other line, it is important to invert the area that is derived, since it is now the area above the line that exists within the sector.

The end points are processed in a slightly different way, as the area is bounded by the starting or ending point. This simply requires the two shapes to be derived using the new bounds. The algorithm currently does not consider the case where the cells are entered multiple times, but will be discussed later on during the filling phase, which will correct any duplication of the areas from the overlap. Note that post correction techniques like this can potentially result in the overflow of values from the expected range, thus must carefully consider the data type used to store the intermediate values.

While optimisation algorithms such as the use of repeated line segments can be implemented with ease, the level of precision often limits the usage, as well as the introduction of overheads in dealing with the edge conditions. Alternate optimisation algorithms will be discussed later to improve the efficiency of the operations that are performed as part of this algorithm.

### 9.3.1.1 Line

Table 9.1: Performance comparison of drawing random lines of length 500 units.

Algorithm	Time (ms)	
	Line	Complete
Bresenham	0.0029	86.48
Wu	0.0037	168.78
Super-sampling (256 random)	2.28	173.16
Super-sampling (256 grid)	32.26	201.88
Area based anti-alias	0.0059	140.94

The results in table 9.1 show that the proposed approach performs reasonably well compared to the other algorithms. The first column represents the time used in the traversal, while the second column includes the time used to modify the cell attributes. Since the anti-aliasing algorithms traverses more cells than Bresenham's algorithm, the overhead is increased in the proposed algorithm as the slope of the line in the first octant equivalent approaches 1.

### 9.3.1.2 Arc

With the two lines traversed, the two open ends of the sector can be joined by an arc with similar intentions to the lines. Traversing the cells on an arc is slightly more complicated than the line, as the vertical and horizontal shifts changes proportionally. Using the circle drawing algorithm like Bresenham's allows for the whole pixel to be found, but it does not allow the exact area covered by the traversal (Bresenham, 1977; Van Aken & Novak, 1985).

Determining the area of occupancy can be done using several different approaches, which can be categorised into two types. The first is the exact approach, where the precise area within the arc is evaluated by determining the horizontal and vertical intersection points and using various geometric shapes to derive the bounded area. The second category is the approximate approach, where certain amount of error is allowed by replacing the arc shape with a simpler shape that allows faster processing.

The first of the exact approach involves splitting and deriving the areas of different geometric shapes within the cell. The cell's occupancy can be represented as the combination of a rectangle, a triangle, and the convex hull of the arc, which is bound by the radius of the sector and the two intersecting points. Only the simple case where the arc intersecting the top and bottom of the same cell will be discussed, as the complex case where there is a horizontal intersect is just an extension using one more primitive shape. The area of the rectangle can be derived using the smaller of the two intersects, while the area of the inner triangle can be evaluated using the difference between the intersecting points. To derive the area of the convex hull of the arc, the area of the triangle portion can be subtracted from the area of the sector. The process and formula for evaluating this area can be seen in figure 9.11 below.  $R$  is the distance from the center of the circle to the intersect point of the cell,  $D$  is the distance between the two intersects and  $\alpha$  is the angle between the two radii.

9.3.1.2 Arc

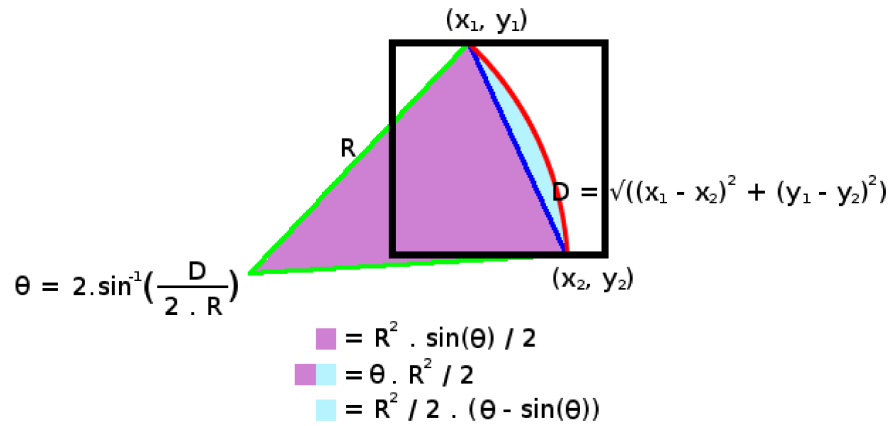


Figure 9.11: Cell coverage derivation using exact points. The green lines show the radii, the red curve is the bound of the circle, while the blue line joins the two intersects to form a triangle with the radii.

The second of the exact approach uses the integration approach by either applying a double integral or splitting the area into sections like before and evaluating the area of each component. Using the double integral approach, the area can be evaluated by assuming unit elevation of the surface. The equations only require the coordinate values of the cell, thus is only complicated by the circle formula to bound the arc. The derivation for the double integral can be found in figure 9.12 below. This process can be visualised as taking the integral over the arced region, then subtracting the rectangular area outside the cell.

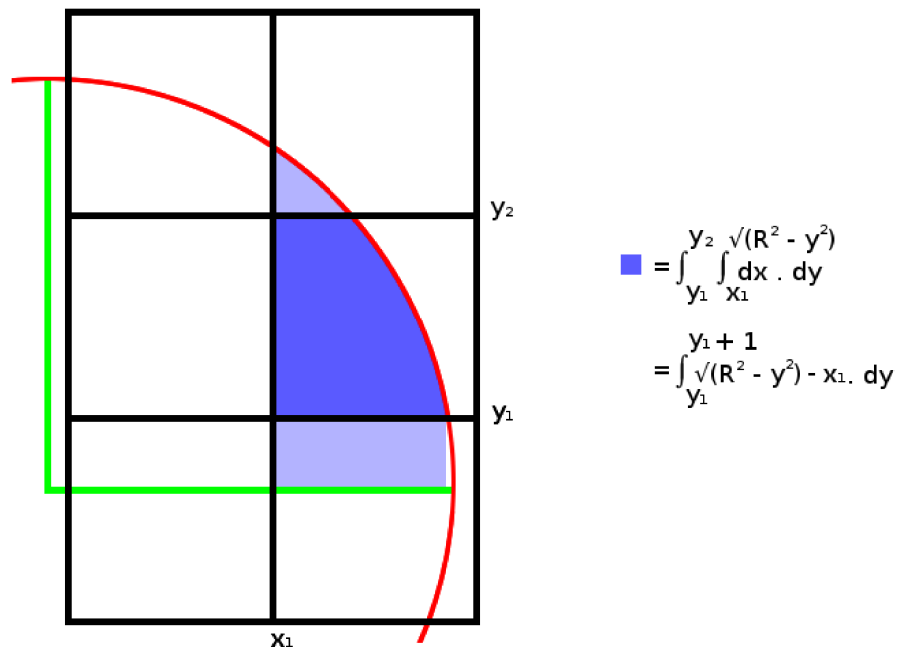


Figure 9.12: Cell coverage using double integral. The green lines show the radius of the circle and the red curve is the bounds of the circle. The blue areas represent the portions where the integration is evaluated over.

### 9.3.1.2 Arc

Although some of the computation can be reused, there are distinct bottlenecks when evaluating the inverse trigonometric functions and the square roots. Although there are obvious signs of slowness, the approaches provide an accurate and consistent time performance. The processing time for the two approaches can be seen in table 9.2 below, where the arcs from sectors with a radius of 500 with a viewing angle of 45 degrees were traced from 1000 different randomly selected positions.

Table 9.2: Performance of the two exact arc drawing algorithm.

Algorithm	Time (ms)
Geometric	0.39
Double integral	0.23

Note that both of the above approaches require all the intercept points to be evaluated. This involves a series of calculation between the starting and ending points to determine the x and y pairs for every cell boundaries. It is beneficial to keep the previous intercept buffered, much like that shown in chapter 6 or to calculate all intercepts first, then use those to iterate through the arc. However, the most time consuming component of the approaches are the trigonometric function calls, especially the inverse functions.

To improve the speed of the algorithm, several approximation algorithms are considered. The evaluation of these approaches consists of the accuracy, processing speed, as well as the appropriateness of the assumptions or constraints placed to achieve the simplification.

The first approach involves the use of the Bresenham's arc drawing algorithm, where the error value is accumulated until it overflows. The absence or occurrence of the overflow determines whether a vertical transition has occurred or there was a diagonal transition. Triggering of the overflow, which is also the amount of error that has accumulated, depends on the cell sizes, thus controlling this can allow for a more precise tracing of the arc. Although the cell's characteristics are determined by other constraints in the whole mapping module, virtual cells can be generated temporarily to allow arbitrary precision for the purpose of evaluating to the desired accuracy.

A critical decision to be made here is the level of precision to be used for the virtual cells. Putting this in the context of a graphic rendering scenario, each of the RGB colour uses 8 bits to distinguish a particular colour. Therefore, the level of anti-alias precision required would be  $2^{-8}$  of the occupancy of the pixel, assuming the cell occupancy is directly proportional to the intensity. Since there are two dimensions to the pixel, the granularity required for each axes would be  $2^{-4}$ . With this in mind, the Bresenham's circle drawing algorithm can be used to traverse the 16 virtual cells within the single cell, as illustrated in figure 9.13. As the traversal is made, the virtual cell positions within the cell can be used to accumulate the overall cell occupancy.

### 9.3.1.2 Arc

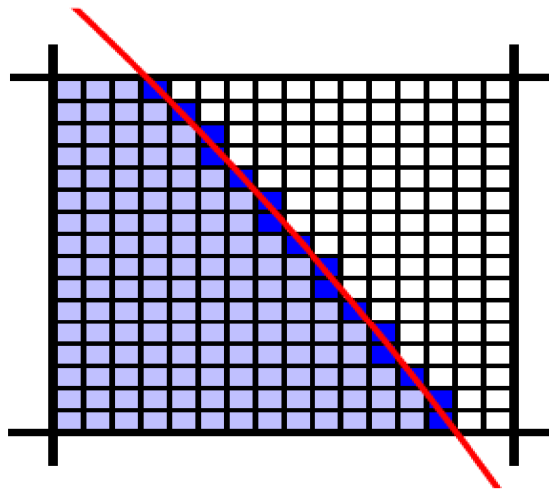


Figure 9.13: Arc traversal using Bresenham's algorithm through virtual cells. The curve represents the bounds of the arc, the dark cells indicate the boundary cells, while the light cells are all accumulated to evaluate the area of coverage.

Although the potential for the improvement in efficiency exists, this approach greatly depends on the chosen precision and the level of errors it is allowed to make. The algorithm is still prone to errors when accumulation of sub-precision values overflow. That is, using the previous example, when the average error in each of the  $2^4$  cells is more than  $2^{-(4+8)}$ . Based on this, the minimum divisions that are required to maintain the same level of precision can be derived. Since the amount of error in each virtual sub-cell must be less than the precision divided by the number of divisions, the minimum value was determined to be  $\text{precision}^{-1}$ . The level of precision that is achieved greatly influences the efficiency, thus the approach is limited to certain situations, where the level of precision can be set quite low. Algorithm 9.4 shows a simplified arc traversal for the first octant using virtual cells.

```
function ArcTraversalVirtualCell(radius, precision):
    set radius = ⌊ (radius * precision) ⌋
    set x = radius
    set error = 0
    set loop = radius / √2
    for y in 0 to loop:
        set sum = 0
        for p in 1 to precision+1:
            set add = x % precision
            set sum = sum + add
            set error = error + 2 * y + 1
            if error > 0:
                if add == 0:
                    carve(x / precision, y / precision, sum /
                        precision2)
                    set sum = p * precision-1
                    set error = error - 2 * x + 1
                    set x = x - 1
                set y = y + 1
        carve(x / precision, (y-1) / precision, sum / precision2)
```

Algorithm 9.4: Arc traversal using virtual sub-cells.

### 9.3.1.2 Arc

The basic idea of integration involves splitting the function into manageable parts and summing the area of each component. By controlling the size of these parts, it is possible to control the accuracy and the speed, as the parts approximate the original function more closely. This technique was applied in the above approach by modifying the vertical and horizontal precision, but a similar approach can be implemented by decomposing the sector into a series of triangles with varying angles from the origin of the sector.

This process is based on decreasing the area of the convex hull of the arc by splitting the cell into two virtual cells, thus introducing another intercept to be used as the vertices of triangle. This allows smaller triangles to form within the convex hull area to better approximate the arc. This process is illustrated in figure 9.14. The two perspectives differ in that the approach using the triangle based at the origin requires a numerous use of trigonometric functions for evaluating the area of the virtual triangles, whereas the introduction of small triangles can be achieved quite easily using the intersection points. As well as being able to specify a fixed number of divisions to be made, the process can be carried out recursively until the area converges to the desired precision which is when the area does not increase by a certain amount.

When considering the granularity of the divisions, it is important to determine the appropriate point to divide the cell to best occupy the convex hull. To identify the best location, the area of the triangles must be evaluated to maximise the coverage. Since the area of the triangle within the sector is maximised when an isosceles triangle is formed, this assists in identifying the optimal split point, as shown in 9.15.  $\theta$  is the angle to the line joining the intersecting points,  $R$  is the radius,  $D$  is the distance between the intersecting points joining line and the arc from the mid-point.

In this particular example, the best location for the split is quite close to the halfway point. Using this fixed value allows for a faster evaluation with reasonable accuracy in specifying the required sub-divisions. However, since this is dependant on the elevation of the cell from the originating point of the sector, the placement of the optimal sub-division point can vary along the arc traversal. Since the arc is evaluated in the first octant, the approximation of the sub-division point to the half-way point is quite reasonable. Algorithm 9.5 below shows a simplified sub-division algorithm with a fixed number of divisions at the half way point.

```
function FindArcAreaTriangleDivision(radius, split, x1, y1, x2,
y2):
  set lower = y2
  set right =  $\sqrt{(\text{radius}^2 - \text{lower}^2)}$  - « x2 »
  set sum = 0
  for i in 0 to split:
    set upper = lower + 1 / split
    set left =  $\sqrt{(\text{radius}^2 - \text{upper}^2)}$  - « x1 »
    set sum = sum + (right + left) / (2 * split)
    set lower = upper
    set right = left
  return sum
```

Algorithm 9.5: Approximate arc tracing using sub-divisions.

### 9.3.1.2 Arc

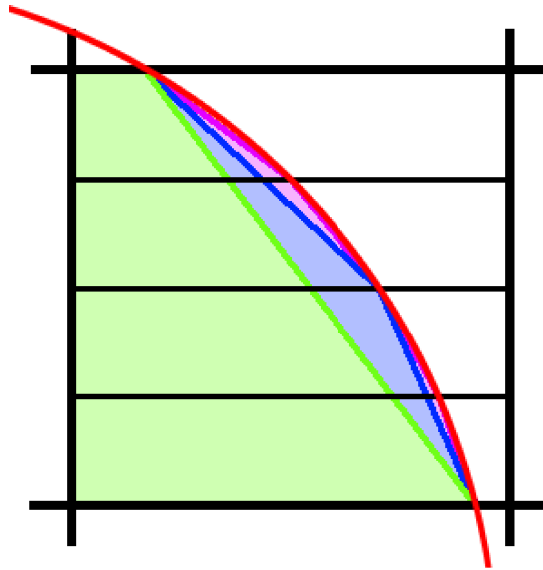


Figure 9.14: Illustration of subdividing the triangle.

The red curve illustrates the bound of the arc, the green line and region shows the base area with no subdivision, the blue line and region shows the extended area being occupied with one subdivision, and the purple region represents the extra area after two subdivisions.

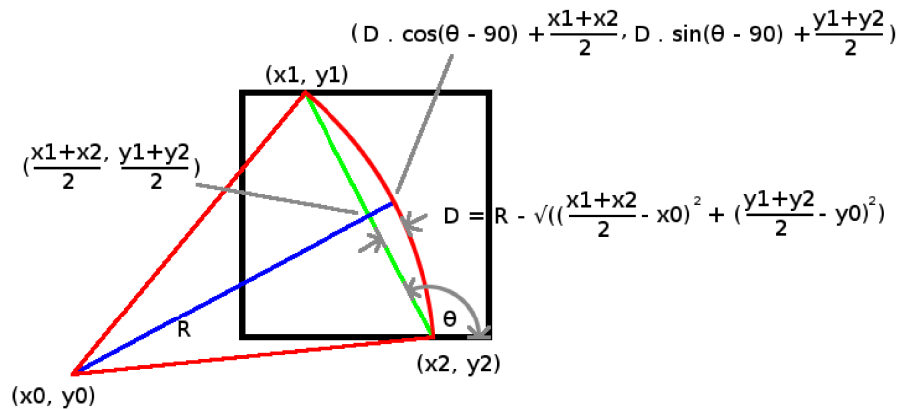


Figure 9.15: Area of coverage at various split points.

The red represents the bounds of the sector, the blue line is the radius which bisects the sector, and the green line joins the two intersects of the cell.

One of the characteristics of the above approach is that the value always converges from one direction, thus will always be less than the desired amount. As observed previously, this amount can be predicted, but doing so efficiently is a difficult task. The error trend can be modelled as a polynomial function, but doing so requires more computation and still consists of slight approximation errors. The derivation of the polynomial function itself is costly, as this depends on the characteristics of the sector, thus cannot achieve a high performance.

Table 9.3 illustrates the two approximation approaches against the Bresenham's arc drawing algorithm to compare their performances. The difference in the accuracy

### 9.3.1.2 Arc

is also shown to note the benefits of the algorithms against the precise implementations.

Table 9.3: Performance of approximate arc drawing algorithms.

Algorithm	Time (ms)	Coverage (%)
Bresenham	0.0089	137.29
Virtual cell (8)	0.84	91.17
Virtual cell (16)	0.97	95.83
Virtual cell (256)	1.62	100.04
Split (1)	0.024	98.89
Split (2)	0.035	99.72
Split (4)	0.077	99.93
Split (8)	0.11	99.98
Split (256)	3.53	99.99

The results indicated that the performance of the approximate algorithms were quite reasonable, especially the splitting algorithm. The small processing time overhead of the algorithm showed very attractive results while maintaining a high level of accuracy.

Similarly to the line drawing, the end points of the arc can be evaluated separately to the arc. Since the anti-aliasing algorithms mark the amount of occupancy in a particular direction, any cell that overlap during the three traversals suffers from excess coverage. This is because the algorithms assume complete coverage of one side of the line or arc. During the traversal, the cells that are accessed are marked to represent the bounds for a filling algorithm. This is done by marking four values for each row or column, parallel to the direction of the filling process. Two of the four are used for the outer boundaries, while the other two are for the inner boundaries. Figure 9.16 shows the various components of the boundary points.

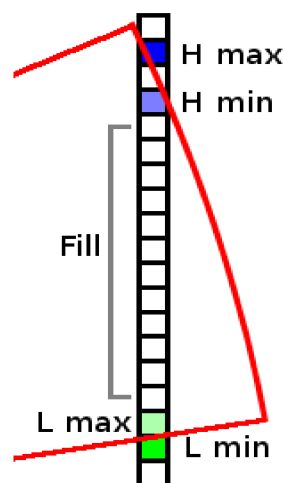


Figure 9.16: Tracking 4 boundary values for the filling algorithm. The four sector intersects are maintained to determine which regions require modification or filling.



### 9.3.1.2 Arc

The purpose of the outer boundaries is to limit the scans when filling, while the region between the inner boundaries indicates whether an overlap has occurred. If the regions do not overlap, the cells in between were not traversed during the definition of the boundary, thus require the scores to be changed as if it was fully covered. However, if the regions do overlap, the cells in between has been visited twice, thus must be adjusted to remove the excess coverage. The excess amount, as shown in figure 9.17, is the size of one cell coverage, thus can be applied quite easily. As for the end points, they require special treatment since the cell coverage excludes certain portions of the cell.

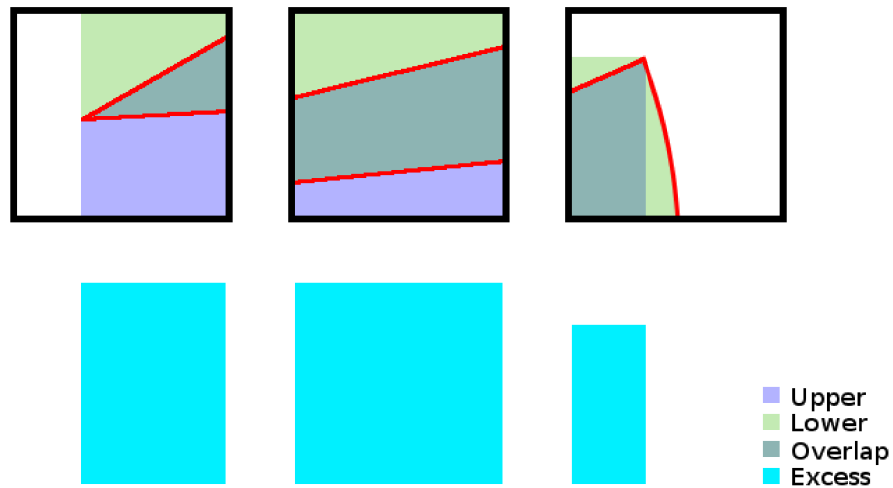


Figure 9.17: Excess coverage from repeated traversal.

The left case shows the modification required at the origin of the sector, the middle shows the case where both the upper and lower bounds of the sector traverse across the same cell, and the right case illustrates modification required where the line bound and the arc bound occur in the same cell.

Since the occupancy is applied to the cells in the arc, it is difficult to note which cells, and how much of the cell, contains an object. It is possible to mark the adjacent cells along the arc as potentially containing objects, but this still does not allow for a clear indication where the object may lie. For this reason, the other cells that are visited are all flagged as vacant based on the proportional coverage of the cell.

### 9.3.2 Optimisation

With the anti-aliased carving algorithm defined, optimisation techniques can be applied to improve its performance even further. As previously mentioned in the arc drawing algorithm, it is sometimes possible to know the required precision before hand. This may come from the granularity of the required result, the precision of the inputs, or a simple constraint defined for the system to prioritise faster processing over accuracy. The limited range in precision can be combined with other weighting values to simulate a higher range, thus allowing short cuts in the carving algorithm.

So far, the algorithms have not specified the data type to be used for the values. The anti-aliasing algorithms that make use of trigonometric or square root functions

### 9.3.2 Optimisation

operate using floating point precision data types, as they have been implemented as generic functions with a wide range of possible values. However, with a specific range of values defined and by using techniques that do not rely on these generic functions, it is possible to make use of fixed point precision data types for faster processing. By knowing the range of the values, the bits can be shifted manually to allow full utilisation of the available bits for the given data type. This then allows the remaining bits to be used to represent the decimal values, thus the approach simply simulates a scaled version of the values so no decimal value is required. Figure 9.18 illustrates several mapping depending on the range of values using an unsigned 16 bit data type.

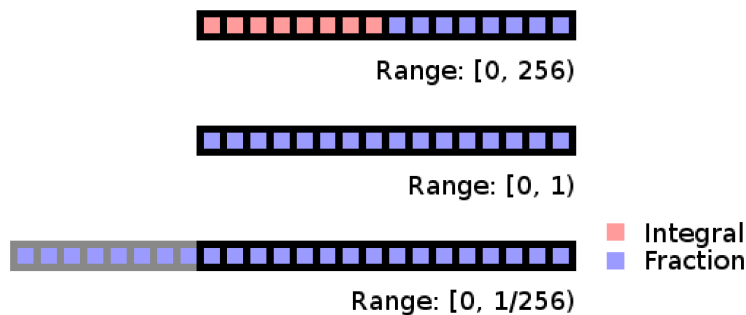


Figure 9.18: Implicit decimal point shifting.

The top example shows 8 bits being used for the fraction component, the middle shows all 16 bits being used for the fraction, while the bottom case shows the case where high order bits are replaced with low order bits.

The applicability of this approach depends upon the frequency of arithmetic operations, the acceptable range of the value, the number of available bits, and the performance overheads in manually shifting the values. Since the range of the cell occupancy has an upper limit, the available bits can be used to represent the low order values. The operations within the carving process are also very frequently repeated, thus improving the utilisation of the proposed approach. The values can then be limited by powers of two's to improve the efficiency in shifting the range, as well as conditional statements. It is possible to derive and use a multiplier or a constant offset, but this management of values introduces unnecessary overheads and negates the benefit of using a fixed point data type.

The shifting and the management before each of the arithmetic operations depend on the operator being used. For an addition, the maximum range doubles and the two operands must be aligned correctly. That is, the implicit decimal point must line up, as the alignment determines the pair of bits being added. For subtraction, the maximum range does not change unless dealing with a negative operand. Unless the sign of the value is known, the operands may require shifting before the subtraction can occur. Since the behaviour of the algorithm is known, as well as the signs of any values that are derived, the operands simply require the adjustment for alignment.

For multiplications, the range can increase dramatically, as the implicit decimal point positions must be added. This often leads to excess bits being introduced which require trimming off, but this happens automatically at the end of the most significant bits, thus the new range must be anticipated and appropriately shifted before the

### 9.3.2 Optimisation

operation. This, however, means that the low order bits are lost in the process. It is possible to select how much of each operand to shift, as long as the resulting value does not overflow. Unlike the addition and subtraction, the two operands do not need to be aligned as each bit is multiplied with the other operand. That said, it is advantageous for both operands to have equal positions, as a biased trimming of just one operand leads to larger errors in the calculation. Figure 9.19 illustrates the workings of the multiplication process.

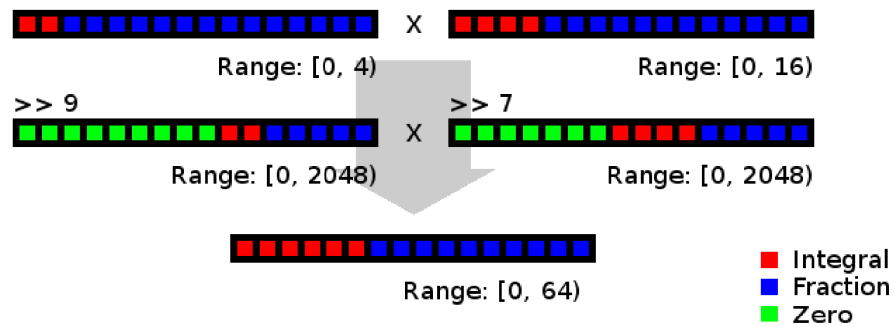


Figure 9.19: Illustration of the multiplication process. The initial shifting forces a loss in the precision to make room for the multiplication to occur and fill in the low order bits from the multiplication result.

The division operator provides the most challenging problem, as there are no clear and reasonable range defined when two operators are divided. Another interesting characteristic is that the divisor must be shifted down such that the result will contain enough bits of information rather than filling up with zero's. This often means a significant portion of the right operand is discarded and the inverse amount is lost from the result. Unlike with multiplication, where the low order bits are populated after the multiplication, the division does not allow for the filling of the high order bits through automated shifting during the arithmetic operation. The inaccuracy introduced by this approach can be ignored if the precision requirement can be set to a very low value, or a larger data type can be used for this operation. Using a larger data type, the dividend can now be shifted up without the loss of information. By shifting the left operand appropriately, the result can later fit back into the original size of the data type. Figure 9.20 illustrates the division process using a larger data type before the operation.

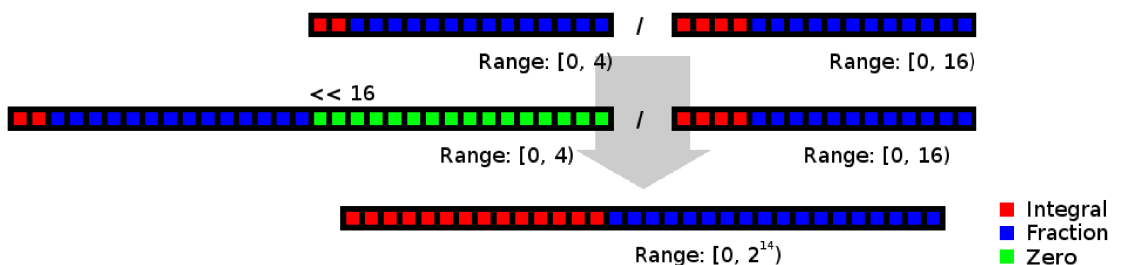


Figure 9.20: Division using a larger data type. The loss in the low order bits during the division means a larger data type is used and padded before the division takes place.

### 9.3.2 Optimisation

These scaled fixed point arithmetic techniques require careful tracking of the implicit decimal point positions and pre-empting of the changes in the range after the arithmetic operations. Since the shifting operations are required before many of the operations, certain blocks of operations can anticipate the overall change in the range and make the appropriate range shifts early on. This can sometimes cause the loss of precision, thus must be used with care so any information that is lost does not have a significant effect on the final result.

Sample code for the shifting technique can be seen below, which is taken from the line drawing algorithm. To simplify the debugging process, the variables are labelled with the position of the decimal point. Note that some of the range is not fully utilised, as operations later on will require the values to be shifted into the appropriate multiple.

```
UINT ui23_x1 = (UINT)(x1 * (1<<23));
UINT ui23_y1 = (UINT)(y1 * (1<<23));
UINT ui23_x2 = (UINT)(x2 * (1<<23));
UINT ui23_y2 = (UINT)(y2 * (1<<23));
UINT ui23_dx = ui23_x1 - ui23_x2;
UINT ui23_dy = ui23_y1 - ui23_y2;
UINT ui30_slope = (UINT)((((ULNG)ui23_dy)<<30) / ui23_dx);
ULNG ul32_inv = (((ULNG)ui23_dx)<<32) / ui23_dy;
UINT ui30_triangle = ui30_slope>>1;
```

Since the approach has limited places where it can be applied, the implementation is currently restricted to the sector carving algorithm. This meant that the floating point values needed to be mapped to and from the fixed point types at the start and end of the algorithm.

The performance of the optimisation is summarised in table 9.4, which show a small amount of improvement for the operations. It may be possible to apply the approach elsewhere, but the difficulty in managing the data means a significant amount of work is required which may not yield any benefits due to the overhead in converting between the floating point numbers, as well as when pre-defined function calls that require floating points are required.

Table 9.4: Performance of manually specifying the decimal points.

Operation	Time (x 10 <sup>-6</sup> ms)		
	Fixed point	Floating point	Shifting
Addition	3.12	4.758	4.086
Subtraction	3.416	5.337	4.492
Multiplication	3.822	7.597	5.226
Division	13.057	27.487	15.584

## 9.4 Global map

The global map, which is used on top of the local map layer, must address four key issues for implementation. The first is the list of attributes to maintain, as the layer will be interested in a different set of information to the local map. The second issue involves the updating of the contents given the local map, as the inputs to this

## 9.4 Global map

layer are no longer directly from the sensors. The consistent interface simplifies the task, but must consider the current state of both maps when being integrated. The third issue is in the management of the attributes representing the environment, specifically in terms of compression when the robot explores an area beyond its initial expectation. The final issue is the use of the map for a large scale analysis. This may include connectivity, path finding, or the detection of anomalies through superimposition of other maps.

### 9.4.1 Attributes

While the global map is largely an amalgamation of many local maps, the differences in the level of detail and viewing area must be reflected by the attributes that are maintained within. Instead of storing the multiple attributes that make up the occupancy in the local map, a single value can be used to represent the occupancy. This allows a reduction in the number of attributes to maintain, as well as allowing a firm value to base various high level algorithms on.

When combining the inconsistent readings of occupancy and vacancy, the regions with vacancy can be flagged as not containing any objects, even if the occupancy has also been flagged. Due to the shape of the sensor signals, the accumulation of the occupancy would have resulted from the overlaps in the arcs just before the surface. With this in mind, the transition between a vacant and non-vacant cell is used to identify the location of objects.

The occupancy is used to distinguish the regions it has already observed, as the occurrence of the transition between vacancy and occupancy indicates that the boundary has been observed. This measure allows the sides of the sector to remain ambiguous until more scans are carried out.

An attribute which has not been well used until now is the surface orientation. This attribute, when combined amongst other cells, can increase the efficiency in forming continuous surfaces of obstacles, as it can guide and validate the possible connectivity between adjacent cells. Since the higher level analysis will deal with the formation of shapes and patterns, the global map must also carry this information to remove the irregular jumps between layers.

Another attribute to be included is the presence of dynamic obstacles. Since the map is built up over a significantly longer period of time than the local map, it is possible to identify changes within the obstacle's arrangement. This includes chairs and doors being moved around, as well as the presence of people that obstruct the sensors temporarily. These obstacles tends to be quite small in comparison to the static obstacles, thus the aliasing effect can hinder the localisation of these objects later on.

The last attribute to be maintained is the ability to traverse through the cell. This attribute is required as the arrangement of the objects within the cell can be lost if the map is compressed. The details of this attributes will be described later.

## 9.4.2 Update

### 9.4.2 Update

Being on a separate layer to the local map, the global map can assume many simplifications provided by the filtering which occurs at the lower level. This includes dealing with noisy sensor readings, uniform size and range of cell attributes, as well as any cleaning-up from higher level analysis carried out using the local information. The updating of the global map requires the timing, the size and also the weighting of the attributes to be considered before the superimposing can occur.

Although synchronisation between the maps is desirable, excessive occurrence results in a large processing load and a reduced purpose of the extra layer, as the environment in the local map is not given enough time to accumulate. The process should take place when the local map has built up enough information, as well as just before certain information is discarded due to the area leaving the viewing area of the map. The first strategy to be considered is a timer based approach, where the updates occur at a specific interval. The second and third approaches both involve analysis of the robot's state in determining the need to update. The simpler approach is done by analysing the amount of change to the local map has occurred since the last update, which can be derived from correlations, as well as accumulating the changes it has observed. The last approach involves the use of a traversal accumulator, where the relative or absolute distance traversed by the robot is used to trigger the update process.

The timer based approach and the accumulation of change approaches have some common grounds when used with particular settings. The rapidly scanning range finders poll and update the local map at regular intervals, thus the number of scans is proportional to any given time period. The difference occurs when the area of each scan is considered, as the presence of obstacles will modify the rate of area covered per scan. To make sure no information is lost, the timer value must be reduced to consider the robot's maximum velocity. While this takes care of most normal motions, it is possible for the robot to move backwards and forwards suddenly, possibly trying to escape from a dead end. In situations like this, the timing interval between cells being introduced then being removed at the border of the map is very small. Therefore, using just the timer would not suffice as this will require too frequent updating. Similarly, accumulating the amount of change is often not enough to handle these tricky cases, although the ability to detect dynamic objects with ease can be quite attractive.

Accumulating the distance traversed behaves slightly differently, in that the maps would not be updated if the robot remains stationary. This characteristics means it is more difficult to detect dynamic objects, as the movement of the robot is required before the map is updated and analysed. By setting a simple threshold for the absolute distance, this approach would encounter similar issues to the above. With a simple modification, the threshold values can be set to each of the four directions to handle the problematic situations of rapid changes in the direction.

Depending on the availability of resources, it is be possible to combine some of the approaches above to increase the utilisation of the available information. The four way distance threshold approach provides the safest approach without excessive updating, thus can form the base condition. The ability to form the global map without motion can yield greater flexibility in the sensors used, such as the sonar

## 9.4.2 Update

sensor on top of the servo motor, which can sweep a large area even while the robot is stationary. These scans typically allow greater precision than those from motions by the robot, and can also be used to track the path of dynamic objects with ease. For this reason, a secondary condition is introduced which accumulates the amount of scans conducted to trigger the update.

Rather than using an area based accumulator, an angular coverage or a simple counter can allow for a more consistent and efficient performance, as the viewing angles of the sensors remains constant. Using this also allows the map to update while the robot stops to decide what to do, such as path finding. Depending on what portion is updated, it can be advantageous to reset both the distance thresholds and number of scan counter when the update occurs to reduce the unnecessary updates.

The frequent merging of the local map can result in excessive copying if clipping of the uninteresting portions are not done. The simplest approach is to copy the entire local map across, which has little overheads in implementing the algorithm and also allows more correlations to occur if the alignment is considered as a reinforcement measure to the pose. Although this allows for the confidence measures to be increased, many of the cells remains unchanged from obstruction, which should not affect the confidence value of the global map, thus are unnecessary.

It is possible to flag the cells when they have been modified since the last update, thus allowing for selective updating. However, individually determining whether it has been modified can consume precious processing time. Since the flagging of the modified cells occur in groups due to the span of sensor scans and the density of the flagged cells are high for areas around the robot, it is possible to make use of template shapes which groups the modified areas. Although the modified cells around the robot will tend to form a circular pattern, traversing this shape can be quite costly unless a change is made to the coordinate scale, such as to polar coordinates. Instead, assumption that the sensors will scan an area consisting of flat surfaces is used to allow the area to be rectangular, thus allowing a simple sequential iteration of the region. A bounding rectangle can be maintained for the sensor scans, which indicates the recently modified region within the local map.

The different conditions which trigger the updating signify the need for analysing the different portions of the map. When transition of the robot is about to lead to the discarding of a portion of the map, only that portion is required to be updated. Similarly, when the number of scans has reached a certain value, it is the areas around the robot that have been affected by the sensors which require updating. By distinguishing the two categories, it is possible to isolate the cells involved and reduce the processing load. Note that the bordering regions are also affected by the bounding rectangle, thus typically only covers a small portion. Figure 9.21 illustrates the two regions that are updated under the scan counter and motion based conditions. Currently, the number of scans used for the threshold is equal to twice the number of angles the sonar is able to point to, which allows for a rough scan of the complete viewing area before the update occurs.

The alignment between the robot and the environment can potentially reduce the number of processed cells within the bounding rectangle if the rectangle is kept at the same orientation as the obstacles in the scene. It is possible to improve this by initially orienting the robot to suit the overall trend in the obstacle's surfaces, or by

## 9.4.2 Update

performing an initial scan to calibrate the initial orientation. The orientation adjustment should be avoided after the robot has started producing the map, as it will require interpolation of the cell attributes to rotate the map or a resetting of the map with the new coordinate system. Since this is just a small optimisation consideration, no great emphasis was placed on these techniques.

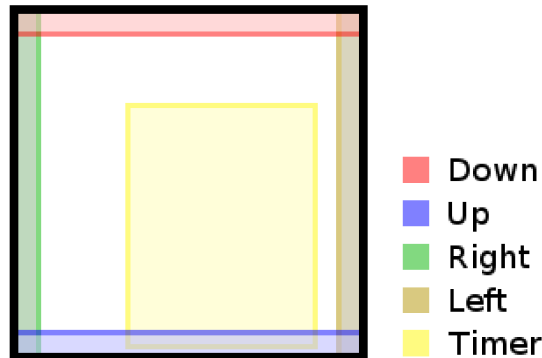


Figure 9.21: Regions within the local map to be combined with the global map. The colour coded regions correspond to the rectangle being copied over to the global map.

With the portions of the local map selected for updating, the cells can be superimposed over the global map with an appropriate weight to merge the information together. Since the local map is intended for portraying the current state of the surrounding environment, the global map should be greatly influenced by the local map. However, the attributes should be treated individually to allow for adaptation in its own way.

The occupancy and vacancy values are both very crucial to the global map, as it contributes to several other attributes. It is important to try and use the up to date values for the map, as the change in the occupancy yields useful information. As noted earlier, the transition between the vacancy and occupancy values allows for the interesting cells to be determined. However, it is the combination of high occupancy and high vacancy transition that yields the desired information about objects. This is used to identify the regions that will increase the interestingness value, while the rest of the regions carved out from the vacancy can be used to reduce the interestingness.

It is important to allow the transition to be noticed, but at the same time, the value should remain within a set range so they can be compared amongst other values within the map. By allowing the three states to be represented using the one value, where a high interestingness is the boundary to the objects, low interestingness is a vacant area, and the midpoint, which represents uncertainty, either from not having visited the cell or when there is confusion with regards to the interestingness. To note the dynamic objects, the interestingness value will transition between high and low very quickly. Another way to look at this is to say that the dynamic object is found if the interestingness moves towards the midpoint. This allows the interestingness value to be adjusted normally to add confidence about the structure of the cell, while the discrepancy in the change to the value can be used to add to the attribute representing the dynamic state of the cell. With this in mind, a weighted averaging can be carried out such that the newer value is given more precedence.



## 9.4.2 Update

In future implementations, it may be possible to make use of a frequency counter to observe the number of scans the cell has received and use that to weigh between the old and new values. In the current implementation, a weight value of 0.75 is used along with the change in the vacancy of the surrounding cells and the occupancy.

$$\text{Interestingness}_{\text{new}} = (\text{Occupancy} - \Delta\text{vacancy} + 1) / 2 \quad (44)$$

$$\text{Interestingness} = \text{Weight} * \text{Interestingness}_{\text{new}} + (1 - \text{Weight}) * \text{Interestingness}_{\text{old}} \quad (45)$$

The change in the interestingness results in the modification of the dynamic obstacle attribute, as a rapid change signifies the presence of a dynamic object, misalignment caused by the incorrect pose, or an incorrect sensor reading. It is possible to detect the misalignment by observing large quantity of the change in the interestingness at the boundary between the vacant and occupied regions. This will be discussed further in chapter 11. Due to the filters provided by other modules, the majority of sensor errors can be ignored. Those that do occur are often detectable during a higher level analysis, such as using the connectivity.

By using the difference in the interestingness as the modifier, the confidence value of dynamic content within the cell can be shifted up or down. One thing to keep in mind is that the change in the occupancy to and from vacancy increases the likeliness, while the lack of change should reduce the probability that the cell contains dynamic obstacles. However, since the movements of these obstacles do not necessarily occur frequently, it is also plausible to simply ignore the case of reduction in the confidence and allow the value to continually increase.

Since this range should be limited, the modifications must be scaled so it does not cause any overflow. By using the magnitude of the change in the interestingness as a proportion of the remaining value, the accumulation can be limited to a specific range. This can be seen in the following formula, where the non-scaled interestingness is used to determine the change.

$$\text{Dynamic} = \text{Dynamic}_{\text{old}} + (1 - \text{Dynamic}_{\text{old}}) * |\text{Interestingness}_{\text{new}} - \text{Interestingness}_{\text{old}}| \quad (46)$$

As the surface orientation measure is simply a rough guide to assist the grouping of cells that represent a surface, this attribute is not as important to maintain with precision. Due to the heavy reliance on sensor scans from a wide range of orientation, the information carried within the cell from the local map is often incomplete and can be misleading. With the sensor scans being reasonably frequent, the surface orientation can be modified to a mask to represent the positions of the neighbouring occupied cells using a binary flag in each of the 8 directions, as shown in figure 9.22.

Although this mask can be derived by simply observing the arrangement of the interesting cells within the global map, the surface orientation can potentially anticipate where the interesting cell will be located, thus allowing confirmations and discrepancies to be evaluated. Although this attribute is not relied upon in the current implementation, it is present for potential future use.

The update process from the local map to the global map can be seen in the following series of diagrams in figure 9.23, which show the updates from transitions and number of scans. The left column illustrates the state of the local map, while the middle and the right shows the global map before and after the update. The top row represents the update from transition, while the bottom row is where the robot stays

## 9.4.2 Update

stationary and the number of scans is used to trigger the update. The interestingness is shown as blue and the dynamic objects in green.

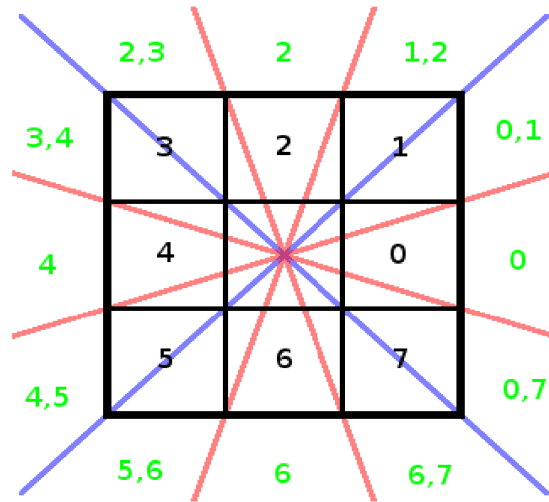


Figure 9.22: Surface orientation to adjacency mask. The green numbers represent the bit numbers being toggled, while the red and blue lines represent the boundaries of the angle and the corresponding bit number.

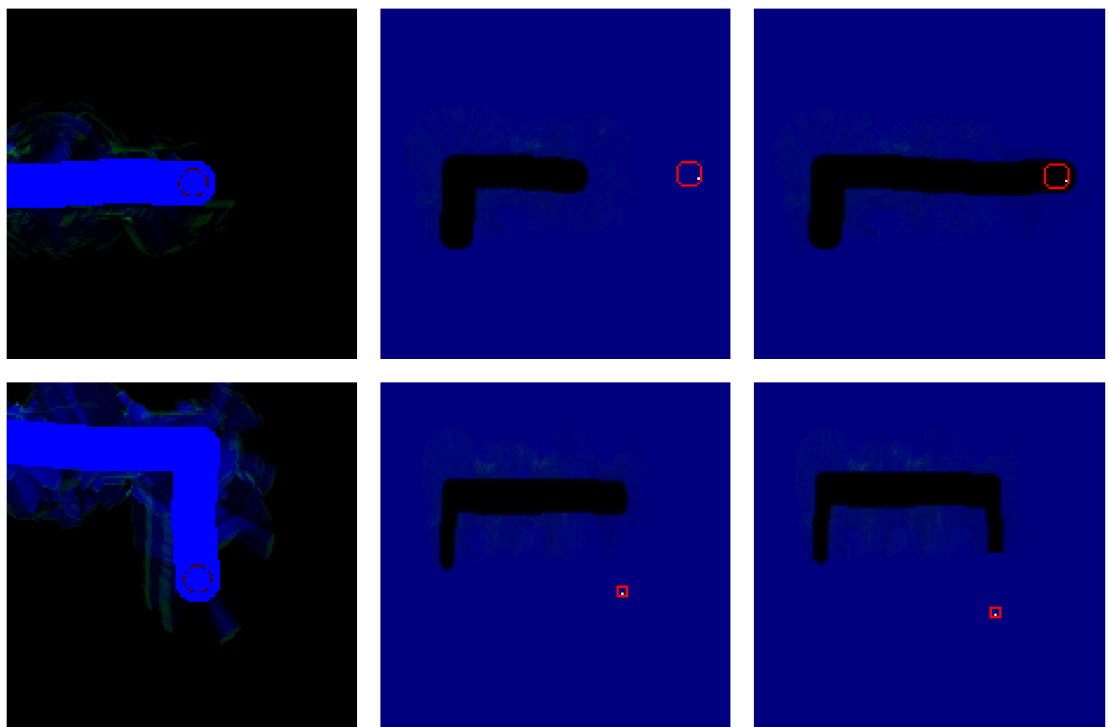


Figure 9.23: Updating of the global map. Top row shows the case where the number of scans has triggered the update, while the bottom shows the update being triggered by transition. The left column is the local map, the middle column is before the update, while the right column shows the global map after the update.

### 9.4.3 Scaling

### 9.4.3 Scaling

As the explored space increases over time, the map must be adjusted to remain within some finite memory space. Although it is possible to anticipate the maximum size of the environment by considering some limitations, such as the battery life and the placement of obstacles to restrict the robot's reach, they rely heavily on the specific environment and a significant amount of domain knowledge. The assumptions also do not provide a way to recover if they are invalidated, nor do they allow for a reasonable bound when multiple maps are combined, as they may dramatically increase the memory requirement.

A simple approach to bound the maximum size of the global map is to modify the scale of each cell when the size of the map becomes too large and compress the neighbouring cells into one. This approach can be efficient to implement, but suffers from data losses as portions of the attributes are lost during the merging process. It is possible to apply a more complex compression algorithm which may allow the data to be retained, such as variations of run length encoding or frequency based algorithms like Hoffman's encoding, but these do not constrain the upper bound to the memory usage and can also consume valuable processing time in maintenance and de-compression when being accessed. It is also difficult to combine multiple attributes for a consistent compression, thus the map may require several layers with irregular factors to associate the various attributes together.

A location dependant compression that occur between the cells means that the various attributes, such as the occupancy and surface orientation, are merged together. This allows for quick access to the appropriate data for each cell, as the locality is maintained. The merging process between the neighbouring cells requires considerations into the weights for the interpolation of the individual attributes as well as the direction of the merge. The attributes can be merged together by approaches like summation, multiplication, averaging, or by taking the maximum or the minimum, depending on which is more suitable for the attribute and the consequence of the resulting value.

For the traversable attribute, the cells must indicate if it is still possible for the robot to move through the cell, thus the merging should not affect the already established path. It is important to note that if the path is blocked by the other cell being merged, the value would not be an accurate measure if the updated value indicated a blocked path. By prioritising the path being not blocked, this value can later be modified if the cell is deemed to be not traversable using a high level analysis, such as a path finding algorithm. Another issue to note is the change in the characteristics if the cell size is too small for the robot to fit inside. In this scenario, it is important to combine the value instead of simply taking the most preferred one, since the cell is never traversable by itself. Multiplying the two values indicates the probability of both cells being traversable, thus is the approach used while the cell size is small.

When combining the occupancy values, the important characteristic to note is whether the combined cells indicate the continuity of the surface. This characteristic can be used to determine if the robot can traverse through the cells, as well as any shape based analysis that may be carried out. Maintaining this information requires a map based on surface structures, which does not fit with the current model of the



### 9.4.3 Scaling

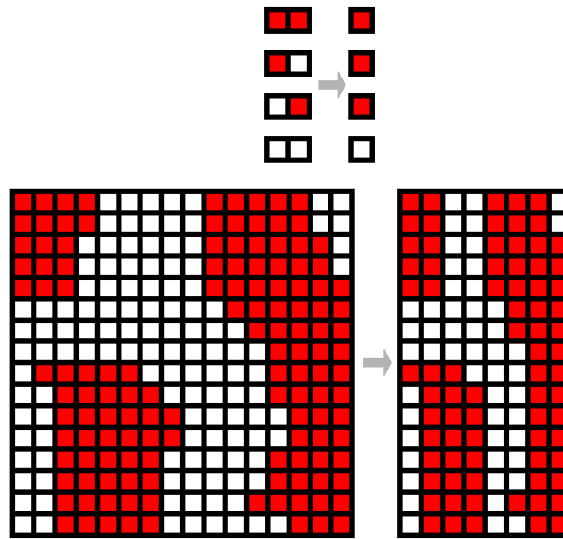


Figure 9.25: Prioritising obstruction to identify paths. Top row shows the compression algorithm used, while the bottom image shows the compression from the map in figure 9.24 to one that is half in width.

The last approach involves the use of two values being tracked for each side of the cell to represent the amount of gap on each of the perpendicular sides. The approach is similar to that of the above, but does not assume a binary state. This allows the cells to maintain the vacancy until the particular side is obstructed, as shown in figure 9.26.

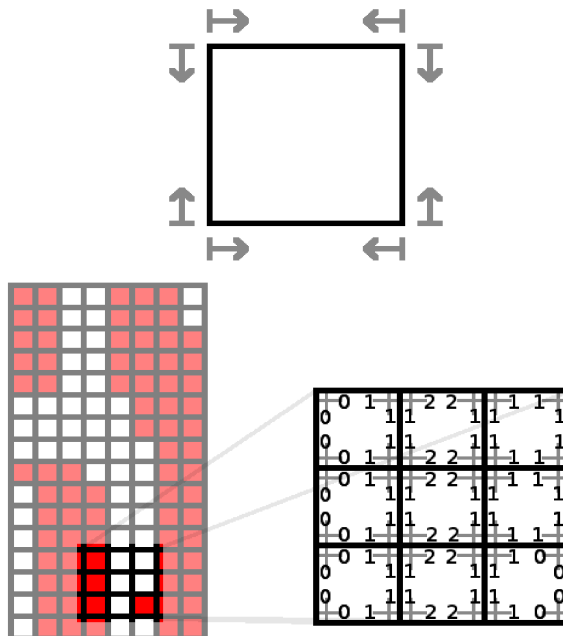


Figure 9.26: Cell merger with tracking of the vacancy along the edge of the cell. The arrows in the top image show the values to track, which indicates the vacancy in that direction. The bottom left is the compressed map from figure 9.25, while the bottom right shows a zoomed version of the map with the vacancy.

### 9.4.3 Scaling

Although the approaches above indicate if the robot can enter and exit the cell, it does not maintain information about the inner connectivity within the cell. This can cause the robot to misbehave as the path that is planned from the global map can potentially be blocked when viewed from the local map. To allow the correct connectivity information, each cell is required to maintain all of the possible paths between the sides of the cell. The lack of distinction in the location of all the possible paths can also lead to certain paths being blocked without recognition unless this too is maintained. Implementing these requires a significant increase in the memory footprint, as well as the processing time to trace the inner paths whenever the map is modified. This has a quadratic relationship with the scale, thus is not suitable in the long run.

It is possible to simplify the approach by finding inner-cell nodes as a form of compression, but a more plausible approach is to not assume that the attribute indicating possible traversal does not mean it is true. By allowing modification of the traversable attribute from path finding algorithms, as mentioned earlier, it is possible to generate a path that can be verified at a higher level when viewed by the local map. If the path is deemed to be blocked, the traversable attribute can simply be changed. This means that the cells that allow any direction of traversing should be marked as potentially traversable, perhaps as a probability value based on the number of sides that are vacant, such that the path finding algorithm is able to rank the various paths it generates.

When considering the attribute for dynamic obstacles, just the acknowledgement that a movable object is present is quite useful. The precise locations of these are difficult to determine due to the time interval required to register the change in the interestingness. Since the aliasing and the change of scale also causes the change in the location, it can be safer for the robot to note the whole area as containing dynamic objects. This can then be used later on to group together the moveable components within the scene, or to be wary of when generating a path of traversal.

As for the hazardous attribute, which is based on four states of unknown, non-hazardous, potential hazard, and hazardous, the critical state to be prioritised are the ones that can potentially harm the robot. When the states are combined, the sequence of states that should be prioritised in order is hazardous, candidate, unknown, then finally non-hazardous.

When the surface orientation is left as an angle, the characteristic is quite different to the other attributes in that the variation in the orientation cannot simply be merged and that the value is cyclic. In the case where the two values are similar, it may be possible to average the two together to cater for the lack of observation from a particular perspective. However, if the two are perpendicular, it could indicate the presence of a corner with two faces. As noted earlier, the surface orientation is not a crucial component, thus has been greatly simplified to the bit mask representation pointing to the position of the neighbouring surface cell. This can be used to allow a simple OR process between the merging cells to indicate the direction to traverse in in order to find the next occupied cell. This process is illustrated in figure 9.27.

The compression factor plays a significant role in deciding how much of the information is discarded, how frequent the process is required, and also how controlled the merger is. The larger the compression factor, the more drastic the

### 9.4.3 Scaling

changes it will cause to the state of the global map. This can introduce ambiguity, misalignments, and potentially allocate large amount of space for places where the robot will not explore. However, this also reduces the frequency of the compression to reduce the processing cost. Although it may be possible to dynamically specify the compression factor, correctly choosing this value requires a reasonable prediction of how far the map will need to be extended.

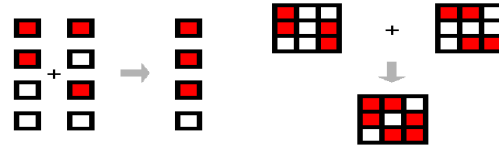


Figure 9.27: Merging of surface orientation mask.

Left image shows the rules used when merging the surface orientation masks, while the right image shows an example merging between two masks.

A possible work-around for the unwanted allocation of space is to detect that the robot cannot navigate to a particular part of the map and shifting the map internally to allocate more memory to a vacant side. It is also possible to split the map with different scales for each portion, but this can introduce complications when accessing the map.

The validation of the above strategies are difficult to do objectively, thus is carried out through the observation of the map, which can be seen in figure 9.28, where the map is compressed by halving the map in one direction. The scaling factor was chosen to simplify the merge process and to avoid the sub-cell interpolation. This also means the attributes are localised if the map is ever expanded again with a different scale factor in the future. The red colour represents the probability of the cell being able to be traversed, the green represents the cells containing dynamic objects, while the blue represents the interestingness of the cell. Note the cell size is still too small for the robot, but rather than representing the eight transition bits, the probability is shown for visual purposes. The green regions are present at the ends of the sectors, which are all the result of slight sensor inaccuracies.

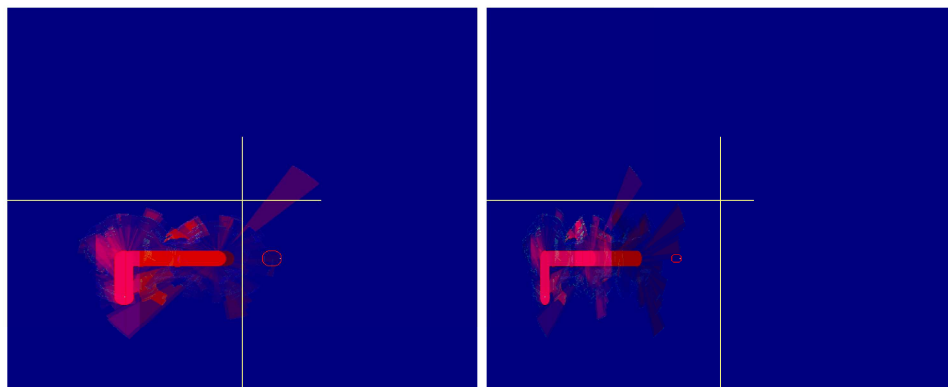


Figure 9.28: Scaling the global map by a factor of 2.

The left image shows the global map before the compression, while the right image shows the compressed version. The yellow lines from the center indicate the scales and also on which side the compression has occurred.

## 9.5 Summary

### 9.5 Summary

The use of the grid map has allowed simplification in the mapping of the components in the environment by providing an efficient and consistent representation to be used between the sensor readings and higher level processes. The lowest level map forms the interface for the sensor reading and maintains five attributes, which are the occupancy, vacancy, number of scans it has encountered, the surface orientation, and the sensor orientation of the last access. The global map, on the other hand, uses the state of the local map to update its four out of five attributes, which are the interestingness of the cell, surface orientation, presence of a dynamic object, and the transitions that are allowed into the cell. The last attribute, which represents the hazardous cells were not discussed in this chapter. Note that to store the transitions allowed for a small cell requires multiple values, while the larger cells only require one, which is the probability value. Once the map grows large enough such that each cell is bigger than the robot, the eight transition values can be discarded for the probability value.

The attributes above add to the simple maps seen in many applications. Although many of the attributes that may be added are specific to the application, the independent maintenance and analysis of each attribute is important in handling the specific information they contain.

To improve the efficiency of the carving process, while catering for the aliasing caused by the use of the grid map, a fast line and arc drawing algorithms is introduced which uses an area based anti-aliasing to determine the bounds of the sensor scans. An optimisation technique is also introduced by specifying the range and precision requirement of the attributes.

The update process encourages both the build up of the local map to contain more information and to maintain an up to date global map. This is done through two conditions to trigger the update, which occurs to a constrained area for efficiency.

The various weights introduced allow the mapping module to operate continuously to represent a model for the current state of the environment without overwhelming the map with redundant attributes. By maintaining a controlled bound to the values, it does not require normalisation, which can be dependant on the trajectory of the robot.

Strategies when the global map expands from extended traversal include the merging of the attributes, such that each attribute is modified to prioritise the safety of the robot and to encourage the re-measurement of the region if ambiguity arises.

As noted earlier, the current implementation does not maintain a permanent record of the local maps as portions are discarded. However this data could be useful for future implementation such that post analysis can be carried out by a more powerful computer with much more memory capacity.

Another potential place for extension is in the use of shape and density based clustering algorithms based on the distribution of the occupancy, vacancy and the dynamic attributes. These high level concepts will allow the formation of topological maps, where the separation of the clusters will allow objects to be segmented and matched against templates to be given a high level tag or characteristics.



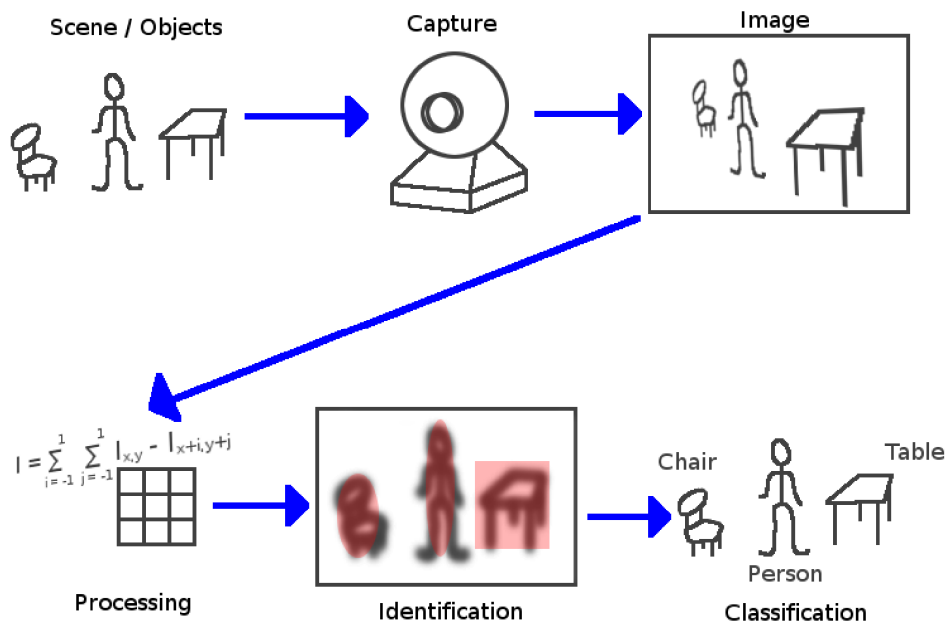
## Chapter 10 – Vision based map enhancements

While the basic map has been constructed using the range finders, the other sensors mounted on the robot can be used to capture and analyse different aspects of the scene to provide an alternate perspective to improve the map. A commonly integrated sensor that to observe the scene is the visual sensor, which allows simultaneous capturing of a region of the scene at a fast capturing rate to allow almost a continuous view of the environment. The availability of large quantities of data from the single device is attractive from the perspective of hardware cost per data generated, while the ability to capture the neighbourhood information allows multitudes of processing algorithms to be developed for the extraction and analysis of the view, such as tracking the motions of specific regions of interest and analysing the inter-feature relationships (Davison, 1998; Zhao, 1998).

Although some aspects of the visual sensors were discussed in earlier chapters, the scenario and the configuration differ significantly as the camera is used to observe the scene looking parallel to the ground. The camera can then observe the scene in a similar fashion to biological systems, as well as allowing wider interaction capabilities with the camera due to the large viewing area.

The general flow of process for a vision based system consists of multiple stages, as shown in figure 10.1, which illustrates the derivation of a semantic representation of the scene defined by the various algorithms used to interpret the image.

Figure 10.1: Flow of process in interpreting the scene.



The scene is observed by the camera, which produces a perspective dependant image. This is then processed to identify the objects of interest and classified by matching the unique characteristics of the objects of interest to some knowledge base.

Many of the approaches introduced earlier to clean up the image can be applied here, thus the focus of the analysis is placed on the conversion of the image data to a representation that can be interpreted, as well as the extraction and handling of any information that are derived from analysis of the pixel patterns.

The current approach for using the visual sensor is based on improving the attributes already found within the local map. This allows the various sensors to support each other instead of introducing more attributes that cannot be verified by other means. There are uses for unique characteristics to be measured, which will be discussed in more details in chapter 11. Although many of the objects that is in the view will appear within the local map, the range of the visual sensor is much larger than the range finders that are used. This can yield new challenges in detecting and using the visual features to improve the map.

### 10.1 Camera configuration

The camera that is used has a similar characteristics to the cameras used for the localisation module, with the exception of the minimum focus distance, which could be set quite small. The camera was originally used to track the ground texture from a very low height within the robot's body, until multiple cameras were required to track the ground. The extra range in the viewable distance does not have a significant impact to the scene analysis, as the focus of the camera cannot be modified without manual intervention and the objects in the view is typically distant from the camera.

One other difference in the camera characteristics is the slight increase in the radial warping effect caused by the curvature of the lens. This meant that the majority of the precision measurements should be carried out as close to the center of the image as possible and minimise the use of the outer regions.

As mentioned earlier, the camera is currently mounted on top of the servo motor, along with the sonar sensors. This allows for a greater control in scanning of the scene and an independent viewing orientation to the robot, which allows a particular point to be continuously viewed while the robot moves around (Ardaiz et al., 2005; Taylor et al., 2006). This ability allows for an effective tracking of dynamic objects, as the camera can modify its orientation to maintain the object within the view. The duplication of the axis of freedom means the camera is unable to observe the scene from a new perspective by changing the pitch and the roll. This limits the viewing of tall objects, such as people's faces, unless the camera is tilted during the initial configuration. By tilting the camera up, the camera will not be able to interact with objects that are close to the ground, which can include hazardous structures like stairs.

Tilting the camera also introduces a transformation between the vertical plane of the environment and the view, thus requires a mapping process between pixels to note the corresponding relationship to the scene structure. Although the transformation process is not a difficult one, nor is it significantly time consuming, many structures in the environment have a boundary that extends straight up from the ground which appear as diagonal lines in a tilted view. When this is viewed through a grid, the aliasing effect causes the alignment of the line to become irregular, depending on how far the view has been rotated and the quality of the camera sensor.

## 10.1 Camera configuration

By assuming that the line detected is perfectly perpendicular to the ground, it may be possible to note the misalignment of the image by observing the trend in the intensities. However, the effectiveness can be hindered by the change in the lighting that causes gradients in the edge's textures, as well as subtle differences in the interpolation amount such as from the warping and imprecise pose of the camera.

With the camera mounted at the front and on top of the servo motors, the depth and length axes are constantly rotated, such that transformations in the intensity arrangement occur. The transformation of the structure has prompted for many image processing algorithms to be developed to identify non-morphing features, such that the same object can be identified in between multiple views (Lowe, 1999). These sometimes include tolerances to the slight intensity changes, such as those from changes in the lighting conditions.

By limiting the motions of the camera, some of the transformations can be eliminated. This can be used to specify a criterion for the feature to reduce the computational load and increase the reliability of the features that are detected, as both the inter-pixel trends and the motions with respect to the change in the view can be anticipated. However, unlike the floor pointing cameras, the number of constraints that can be applied is very limited, thus requires sophisticated algorithms or additional sensor readings to disambiguate the causes of various changes to the view. The visual sensors are used to support and improve the state of the map by identifying the surfaces and the position of corners where the depth changes suddenly.

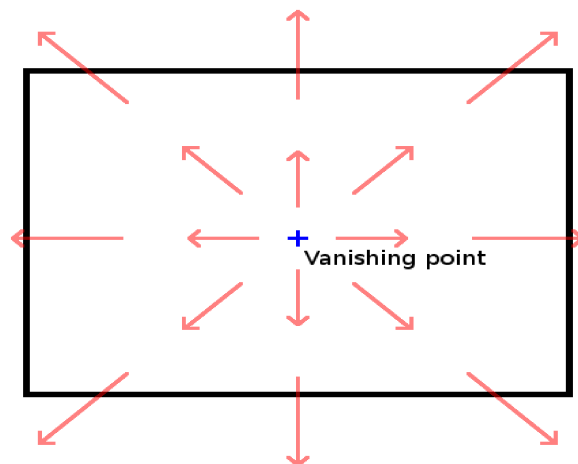


Figure 10.2: Feature motion within the view under forward motion by the robot. The red arrows indicate the motion of static objects as the camera is moved forwards.

One potential drawback to mounting the camera at the front is the reduced displacement of the features, as the majority of the motion will be a forward or backward motion by the robot. Figure 10.2 illustrates the motions observed by the camera at various orientations when the robot moves forward. A motion along the depth axis results in the combination of vertical and horizontal motion within the view, which causes an increase in the search area for features between the frames, as well as introducing more ambiguity through increased dimensions of sub-pixel shifts.

## 10.1 Camera configuration

It is possible to increase the viewing area using the servo motor, but the inter-frame information does not provide much useful pose information, as the rotational point is almost, if not exactly, the same as the focal point. Since the camera sensors determine the orientation of the incoming information, the rotation of the image plane simply shifts the relative orientations. This is useful in tracking the motion of a dynamic object or when a larger scene needs to be scanned, as it maintains the neighbourhood information between the objects to simplify the correlation process between the backgrounds for image stitching (Brown & Lowe, 2003).

By facing the camera to a perpendicular direction to the traversal, the forward and backward motion by the robot will constrain the motion of the view to a horizontally dominant one. This can lead to a much more reliable tracking of features to note various characteristics about the feature. Since the maximum rotational angle for the servo has been set to  $\pm 60$  degrees and the constraints by the differential drive system, the camera is purposely rotated by 30 degrees to be in an ideal orientation when the servo is set at the maximum anti-clockwise rotation. In future implementations, the camera should be given an individual motor for pitch and yaw control, to allow the sonar sensor to scan the area freely while the camera observed the sides when traversing and is able to change its view to focus on different targets when desired.

## 10.2 Image processing filters

The extraction of the scene structure from the captured image first involves the filtering of any noise that may have been introduced by the environment and the sensor. The major difference between the process used here and in the floor pointing cameras is the reduced constraints that can be applied to the observed scene. The camera sensor requires more filters to remove the additional noise and ambiguity in the information it captures.

As previously, the image that is captured uses the full colours that are available at the maximum frame rate, thus allowing the individual algorithms to select the components they desire. Using the currently viewable image, the filters are implemented to observe both the current view and the transition between the previous frames.

### 10.2.1 Ambient light

One of the most difficult issues with using a passive sensor is the reliance on the ambient light and the need for filters to suppress the variety of changes in the appearance of the same object. In many image processing tasks, the colour information is quickly discarded and replaced with shape based descriptors, such as edges and corners. Although this often allows for an ambient light independent view of the scene, it also discards potentially useful characteristics about the object, as well as the state of the environment. By successfully monitoring the changes in the ambient lighting conditions, the attributes or the criteria for the features can be modified to increase its reliability.

The ambient light changes can occur as gradual or sudden change to the scene, such as from shadows and illumination. As noted earlier, the exposure rate and the

### 10.2.1 Ambient light

colour balance used by the camera can modify the apparent light conditions to regulate the overall brightness of the image. Although this feature can be useful, the specifics of the adjustments are not always available for use and the enhancements that are performed does not guarantee that the intensity characteristics for a particular object remains the consistent. However, if the various camera settings are set to a fixed value to start off with, the changes to the ambient light conditions could disable the camera's functionality as the whole image might appear as a saturated white or be too dark and noisy from unforeseen conditions.

As noted earlier, the goal of the visual sensor is to enhance the attributes and precision of the local map, thus the ability to continuously track a specific feature for a long period is not as crucial as accurately tracking with respect to the pose of the robot. With the exception of dynamic objects, the features that are observed by the camera does not have to be tracked immediately, as the object will still remain in the same pose when the robot returns to view the area again. This means the object can still be found again even if the feature is lost. However, it is still an attractive ability to be able to track the feature for as long as it can, as the continuity considerations can be greatly simplified and the changes in the appearance can also be monitored to note the lighting conditions.

The derivation of the thresholds to differentiate the light condition changes and an actual intensity change depends greatly on the initial intensity, the source of the light change, and the general trend in the intensity change throughout the image. In an indoor situation, there are multiple light sources that interact with the objects. This can contribute to the change in the appearance when the camera observes the object from different angles or at different moments. Since the images are captured with very little time interval, the change in the intensity from any gradual changes can be catered for, as the difference in the intensity will be quite small between the frames. However, for specular surfaces and large changes to the light, such as from elimination or addition of light sources, the intensity change is often too large to be reliably linked with the previous frame by simply relying on the intensity.

Although the use of shapes and inter-pixel trends can be used to support the feature identification and tracking process, an alternate colour scale can be used to isolate the lighting dependant appearance by representing the three colours as relative values.

### 10.2.2 Hue analysis

In the presence of white based light, the hue value remains reasonably consistent under various levels of intensity, even when shades form. This allows for a lighting independent classification of an object texture and is often used to identify surfaces, as the gradient caused by the shade typically hinders a RGB based clustering. At times, it is used in conjunction with the saturation values, but this too suffers from lighting changes and can show gradients along a flat surface.

One of the difficulties with relying on the hue value is the varying sensitivity at different saturation levels, which can cause the hue to fluctuate dramatically for colours nearing grey. This makes any shading removal difficult to achieve on grey scaled portions, thus the techniques must be avoided on these regions. When

### 10.2.2 Hue analysis

encountering a grey portion of the image, the saturation value can be used to determine whether the resulting hue value is reliable or not through a simple weight. Another problematic area includes glossy surfaces, which form sudden fluctuations in the surface texture and does not maintain the same hue value as the intensity saturates the sensor to be able to measure an accurate proportional colour. To account for these, the change in the intensity can sometimes be too rapid to note any trends, thus must anticipate these false positive intensity fluctuations and allow the map to filter them out. As their position is dependant on the pose of the observer, these can be identified by the unusual motion pattern compared to other features located in the vicinity.

When considering the weights to apply to the hue value, it must be accompanied with another value to weight between. Instead of just using the hue scale, the intensity patterns found on the RGB scale can also be used to selectively interpolate between the two ways of observing the scene. Since the rate of change for the saturation is non-linear, as shown in figure 10.3, it may also be feasible to simply use the difference between the maximum and minimum intensity values as the weight.

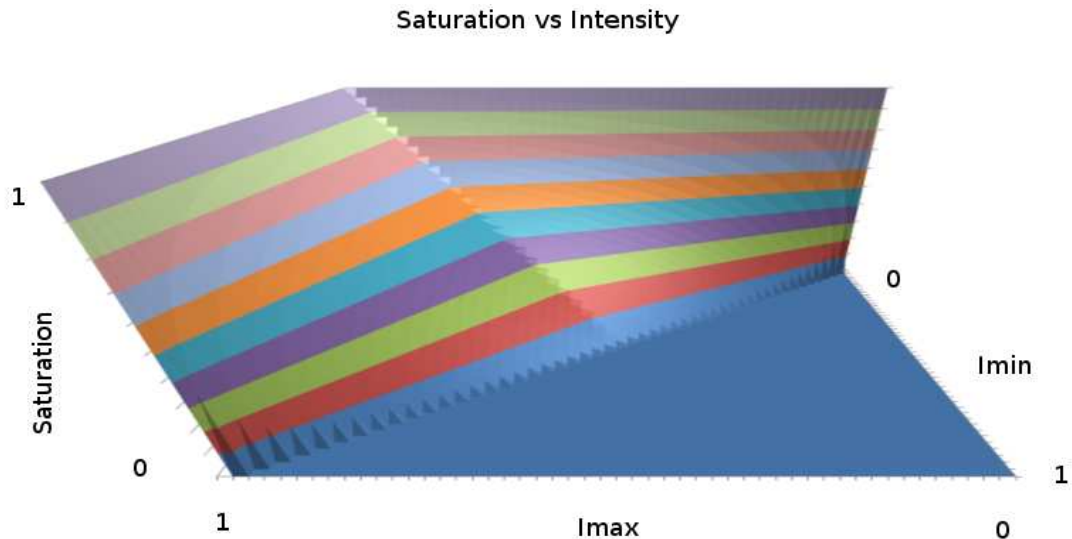


Figure 10.3: Saturation against minimum and maximum intensity values. The rate of change in the saturation depends on both the minimum and maximum intensity.

Since the point of interest lies where there are changes in the image texture, an edge image can be formed, which combines both the hue and RGB based edges. Although it is possible to determine various orientation and positioned edges, the algorithm will focus on horizontal edges, as they allow the borders of surfaces that line up with the map to be determined. When evaluating the RGB edge value, the average value between the three colours will allow the overall differences to be observed, the maximum difference will only focus on whether an identifiable edge was present, while the minimum difference will indicate how reliable the edge is under different lighting. Since the purpose of the edge is to determine the presence of the edges and not on being able to rank the edges based on the overall change in the intensity, the maximum difference between the three colours is used. It is also assumed that the colour of the light source does not differ greatly from the white

### 10.2.2 Hue analysis

colour. Is it possible to make use of the second order differential to pin-point the location where the maximum transition occurs, but this will often create a double edge or the value will not indicate the strength of the edge. The precision that is achieved through the discrete sized pixels and the blurring of the intensities also does not allow accurate selection of the transition point, thus will not be used here.

As for the cyclic nature of the hue value, it is the minimum distance between the two hue colours that are used, as they indicate how similar the two values are. Since this halves the range, it is important to not lose the precision of the hue values from initially mapping the value onto a discrete scale. The transition strength can be derived through a formula like the following.

$$\text{hue\_edge}_i = 1 - |1 - |\text{hue}_i - \text{hue}_{i+1}| / \text{PI}| \quad (47)$$

One of the issues in edge based analysis is the blending in the intensities that occur from the aliasing. The combination of multiple intensities causes the hue to change erratically, thus the edge map that is formed often has a double edge, as shown in figure 10.4.

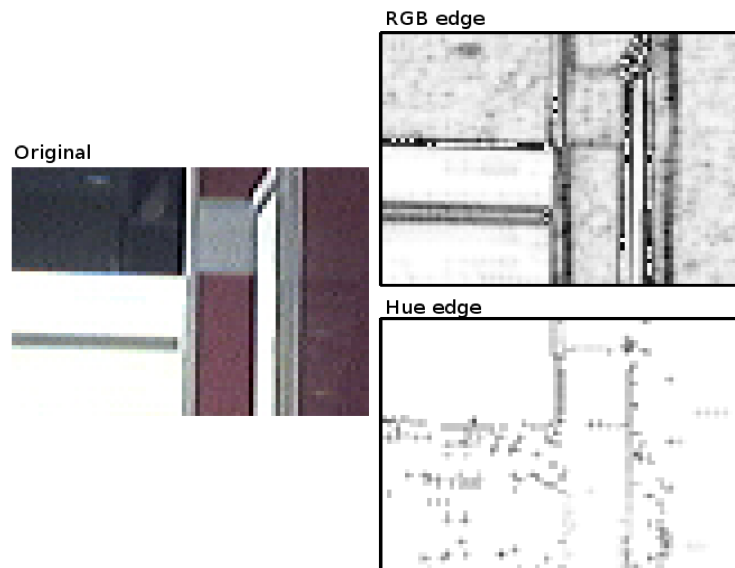


Figure 10.4: Sample edge map showing a double edge. The left shows the original image, top right shows the RGB edge using the maximum difference in the three colours, while the bottom right shows the hue edge, which has not been effective in the office environment due to the lack of multiple hues on furniture.

By combining the RGB edge and the hue edge in various ways, the intended locations of the edge can be derived. Figure 10.5 illustrates several of these approaches. By multiplying the two edges together, it allows the suppression of false positives, but typically scales the strength of the edges back significantly which requires normalisation. Through averaging, the two values can allow the reliable edges to show, but it does not account for the differences in the two values and can often register false positives. By always using the lower of the two, the majority of the false edges can be suppressed at the cost of a reduction in the total number of features that are found.

### 10.2.2 Hue analysis

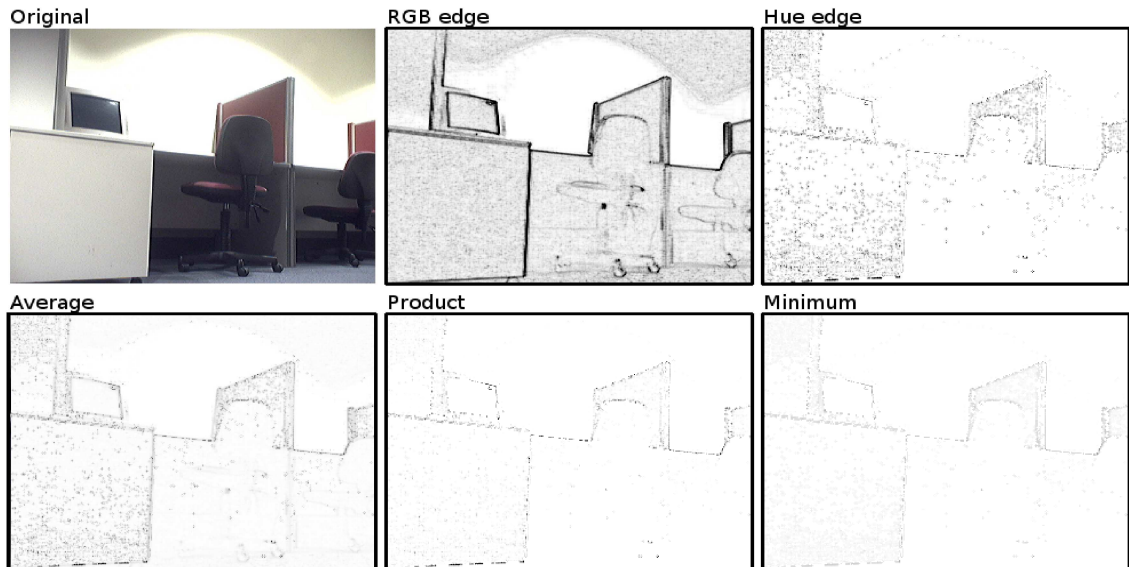


Figure 10.5: Merging between RGB and hue based edges. The top row illustrates the original and the two edge images, while the bottom row illustrates various techniques used to combine the two edge images.

Since the desired features have distinctive intensity differences with the neighbours, the early suppression of insignificant features will save precious processing time later on to remove many of the non-distinctive feature candidates. The increase of the type II errors that is introduced from this should not be a significant issue, as long as a reasonable number of edges that cover the various objects within the scene are maintained.

### 10.2.3 Noise reduction

Linear convolution filters and temporal filters have been commonly used to reduce the level of noise by interpolating the intensity, but suffers from delocalisation of features, suppression of infrequent or small objects, as well as blurring in both spatial and temporal domains.

By knowing the components to look for within the image, it is possible to use these constraints to recover from the imprecision introduced by the filters. These techniques include finding the maximum correlation scores when the template is superimposed over the image at various positions, which can potentially lead to a more precise position of the feature being identified. Without the precise knowledge about the object to look for, or the transformational changes that have occurred between the frames, this approach does not allow recovery of the blurring errors. Instead, the attributes for the constraints are typically derived from the stream of images dynamically by setting approximate bounds and narrowing down to a precise value over time.

Instead of attempting to blindly remove all intensity patterns humans would classify as noise, the purpose of the noise reduction filters must be kept in mind. Since it is the correlation between the features that are found and those present in the grid map that is important, the noise reduction filter can actually be applied after the



### 10.2.3 Noise reduction

features have been found. This allows the filters to simply discard those that are deemed to be generated from noise. Although in doing so, the processing load must be observed such that the management of the features including false features do not consume a significant amount of resources.

In terms of features, there are classifications that are given to distinguish between tracked features, new features, as well as lost features, which will be discussed in detail later. Any noise influencing the tracked features can simply make use of correlation scores to determine what effect the noise has had. Although the initial correlation to a feature that is being tracked may suffer, the change in the intensity pattern can be noted to characterise the noise, whether it was sensor generated, due to the change in the lighting, or from aliasing and interpolations that may have blended the neighbouring pixels.

For the newly introduced features, comparisons with historical information are not available to assist in the elimination of false features. However, it is possible to use the other features that are currently maintained or are simultaneously being introduced to compare the confidence score of the feature being appropriate. One thing to note for small features is the difficulty in correlating with the grid map, as they often do not form a part of the scene. For distant objects that appear small, these map to an area outside that of the local map, thus can be ignored until robot is nearer. With this in mind, the feature should be of a reasonable size to allow the inclusion of neighbours and to utilise the wider range in the score.

As for the features that are lost, the current implementation does not recover from temporary losses due to fluctuations and obstruction. Although this may cause many features to be lost, it prevents erroneous features from being maintained and removes the need for anticipating the motions and transformations of invisible features.

Although the above approaches are all integrated with the feature detection process, there are two filters that are implemented with the specific purpose of removing the erroneous intensity patterns. As introduced previously, the block formation due to the video codec can be reduced using the algorithm introduced earlier. This allows the edge based algorithms to operate without picking up the artefacts as edges, as shown in figure 10.6.

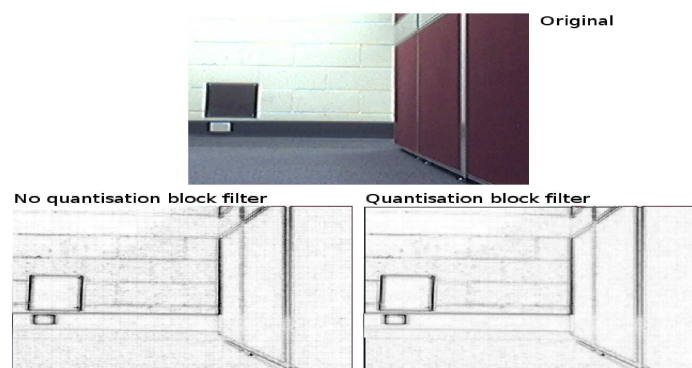


Figure 10.6: RGB edges with and without quantisation block filter. The top image is the original, bottom row shows the RGB image with and without the quantisation block filter introduced in chapter 5.

### 10.2.3 Noise reduction

The second filter that is implemented is dependant on the driver level algorithm used by the camera, in that it monitors the effect of motion blur that occurs due to the automated colour adjustments that are enabled, as shown in figure 10.7. By allowing the driver software to automatically adjust attributes like the exposure time, brightness and white colour balances, the camera is able to adapt to various environments without manually modifying the settings. Using some camera drivers, the current settings can be monitored or manipulated in real time, but in the absence, the overall amount of intensity and the edge image can be monitored to note a sudden change in the clarity of the image. Although this can be done by observing the trend of the scores for all the features that are tracked, the process can be bundled with other whole image processing tasks, such as the copying of the image buffer.



Figure 10.7: Motion blur when the exposure time has been set too high under low lighting conditions.

The blurring, doubling of objects edges, as well as the overlap of intensities is introduced to complicate the image analysis if not filtered out.

If there is a sudden drop in the overall edge score, indicating a large change to the environment or motion blur, the reliability of the feature decreases significantly as the faint streaks and the repeated image that is visible can be incorrectly matched. Instead of attempting to make use of the blurry image, the frame should be tagged as misleading and the feature tracking algorithms should take this into consideration.

The threshold for triggering this case depends greatly on the typical level of fluctuations, the brightness level, as well as the velocity of the camera. After quick experimentation, it was observed that the edge scores can both increase and decrease when the camera motion is carried out at different speeds. This was mainly due to the aliasing between the neighbouring photo-sensors and not from the motion blur. The slow movements specified by both the locomotive components and the localisation module meant only the rotation and the presence of dynamic objects would cause motion blur. To cater for the blurring when the camera is rotated, the command to move rotate the camera or the robot triggers a flag to notify that the features may be corrupted by motion blur, thus should not attempt to use the most recent state to modify any scores. As for the dynamic objects, their presence is assessed in a separate process, which will be discussed in chapter 11.

When observing an object with closely interlaced texture patterns, the sampling

### 10.2.3 Noise reduction

rate of the sensor can sometimes cause a non-existent pattern to emerge. These Moiré patterns are typically removed by frequency based filters or suppressing them through interpolation, but can be isolated as these patterns are typically formed from highly dense regions of intensity fluctuations. This means that the feature selection criteria can observe the surrounding intensity fluctuations to flag that the feature is not distinctive. Even if the pattern is selected as a feature, the motions of the feature will not correspond with the anticipated motions. As this information can be derived from the motions of other features and the localisation algorithm, these false features can be eliminated quickly.

The most problematic artefact that is present on the captured image is the aliasing and the interpolation of neighbouring pixel intensities that occur. As observed earlier, these typically contribute to multiple edge regions being observed, when there is only one. Rather than attempting to recover from this error, the feature selection process can be configured to account for the gradual transition in the intensity measurement.

## 10.3 Features

The significance of a feature is dependant on its distinguishable attributes, the portrayal of a significant region within the local map, and the ability to track its inter-frame motions after changes in the pose. The approaches that are typically applied include the identification of non-morphing characteristics or colour based tagging across a small region of space. These analyses often use the immediate neighbours to observe the relative intensities, which are then compared to other candidates to select the most interesting features within the view.

Depending on how reliable the features need to be, the attributes and the selection criteria can be modified to suit the level of requirement. This often means short-cuts can be made for short term features when determining the most appropriate feature, as well as the rough elimination of some feature candidates to reduce the processing load. As the constraints to the camera motion restricts the motions of the features, only a simple search is required to successfully identify the same feature in the following frame. As noted earlier, the horizontal edges allow the edges of objects to correspond to the change in the depths that is detectable by the range finders.

As mentioned earlier, the feature will be determined through simple RGB and hue based transitions, but must consider several issues to be able to filter out the redundant and insignificant features, as well as any noise which may appear. Instead of attempting to determine the most unique feature, it is more important to identify a group of features that correspond to the same object, thus allowing the acknowledgement of different surfaces. The reason for the simple criteria used to identify the feature is the level of constraint that can be applied to reduce the complexity of the transformations of the feature.

One of the techniques that is used to improve the efficiency of the feature identification is in early elimination of previously processed areas that have not undergone any change. This approach is achieved by observing the difference in the intensity over time through a temporal difference filter, which allows the detection of areas that have changed since the last frame. When the camera undergoes a very small amount of motion, the regions with differences in the intensity are highlighted

### 10.3 Features

as the intensity on one side of the edge shifts to another location. The areas that have not undergone changes can simply be discarded, as no new information will be derived from it. It is possible to distinguish between the background to foreground using this approach, as some distant objects may not show signs of motion. The detection of dynamic objects are also possible if the background stays stationary as the outline of the dynamic object can be determined with ease.

When using this filter, it is important to apply an appropriate threshold value to determine if the change in the intensity is due to a motion within the view or if it was generated by sensor noise or Moiré patterns. The simplest approach is to use a single threshold value, which is taken from the noise characteristic determined earlier, but can also be a more elaborate process such as observing the change in the neighbours. However, since this would require valuable processing time, a threshold value was set using the larger standard deviation of 8 units for the three colours to suppress the noise and unmodified regions.

One of the potential issues with this filter, as discussed earlier, is that under very slow and gradual motion, the intensity change will be too subtle to be detected due to the aliasing and blurring of the intensity between pixels. However, the low threshold value used showed it still allowed for the changes to appear under normal operation speed of the robot. Note that this is used to reduce the candidates for new features and will not affect the tracking of existing features.

A sample scene with portions that have been detected as having undergone change between frames is shown in figure 10.8, where the intensity represents the magnitude of the maximum amount of change in the RGB values. The majority of the image can quickly be discarded as they do not introduce changes to the features.

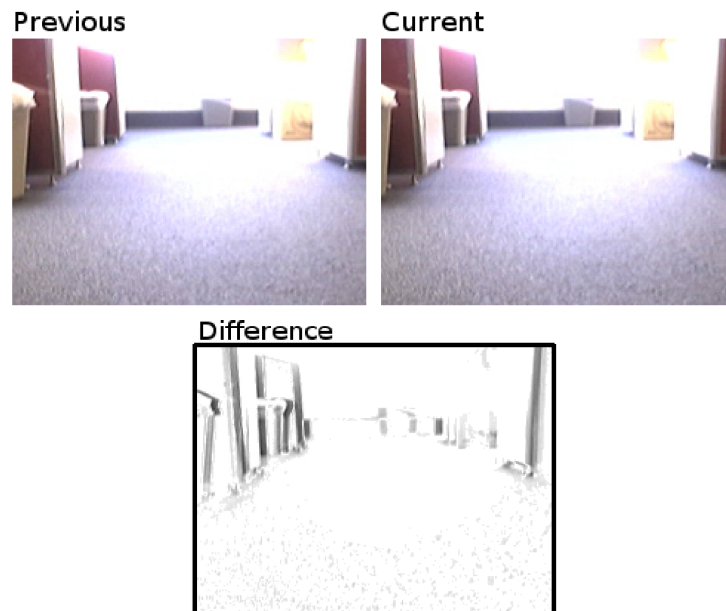


Figure 10.8: Reduction of candidates through a temporal filter. The change in the perspective only modifies some portions of the image, which is illustrated by the dark areas in the bottom row.

### 10.3.1 Feature selection

#### 10.3.1 Feature selection

Due to the repetitive nature of some texture patterns, using a small number of pixels for the texture can be extremely difficult and often leads to incorrect matches and large fluctuations in the correlation scores. However, by selecting a feature that is too large, the objects within the feature may change differently and modify the structure within the feature area. As noted earlier, the main point of interest is the transition region between the intensities, thus storing large number of pixels does not lead to much enhancement of the feature's reliability. However, it is possible to note the neighbours of the edge transitions to isolate a feature from noise (Cafforio et al., 1997).

Rather than varying the shape of the feature to find the most appropriate one like earlier, the characteristics of the objects of interest is used to set the criteria for selecting and maintaining the feature. When analysing the image for a feature, it must involve the pixels covering the boundary to measure the distinctness of the transition, the horizontal neighbours of the boundary pixels so the consistency of the surface can be taken into account, and the pixels above to measure the continuity of the edge so it can be distinguished amongst noise.

When the transition point between objects is observed, the horizontal transitions are typically visible as a combination of two adjacent transition values. If a non-horizontal transition is viewed, the number of pixels being intersected increases to create thicker edges, which are illustrated in figure 10.9. Although it is desirable to encounter the perfectly horizontal transitions, there are frequent cases of misalignment which may cause a slight slanting of the vertical lines.

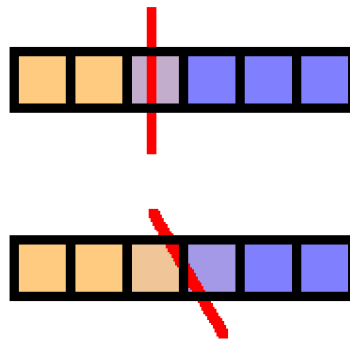


Figure 10.9: Vertical and non-vertical lines intersecting the pixels. The top row shows the object boundary intersecting one pixel, while the bottom row shows two pixels being intersected, thus blending the appearance of both pixels.

Since the constraint is placed to only monitor vertical lines, which are the potential boundaries between objects at different depth, the number of pixels considered for the boundary is kept small. To account for the misalignments, four pixels are used in case the transition occurs across multiple pixels. When the amount of difference is being measured, it is important to monitor the consistency in the intensity transitions, as different objects may also appear within the window for an extra edge. Since the intensity of the boundary pixels tends to interpolate with the neighbours, the pixel intensity in between should be between the two values. That is,

### 10.3.1 Feature selection

the values for the three colour components should lie between those of the outer pixels and should also slope in the same direction.

Although it is the magnitudes of the transitions that are typically used, using this approach requires the maintenance of a signed difference between the neighbouring pixels. It is also beneficial to observe the trends of all three colours separately, since the three colour components can fluctuate at different rates. Instead of a rigorous approach, this can be simplified by determining the difference between the edge score of the transition and the difference between the left most and right most pixels of the window. Since the edge score uses the minimum of the two colour scale differences, it means the value prioritises the lack of an intensity difference. This means more areas will be deemed to be uniform and will suppress some of the intensity transitions. By using the maximum difference of the two colour scales, it sets an upper limit to the difference, thus can be used to determine the overall transition between the pixels. This weight, which is defined below, can be applied to the score of the boundary, as it determines the proportionality of the transitions within the horizontally aligned pixels.

$$\text{Consistency weight}_{x,y} = \frac{\max(|\text{RGB}_{x-1,y} - \text{RGB}_{x+2,y}|, |\text{Hue}_{x-1,y} - \text{Hue}_{x+2,y}|)}{(\max(\text{Horizontal } \Delta\text{RGB}_{x-1,y}, \text{Horizontal } \Delta\text{Hue}_{x-1,y}) + \max(\text{Horizontal } \Delta\text{RGB}_{x,y}, \text{Horizontal } \Delta\text{Hue}_{x,y}) + \max(\text{Horizontal } \Delta\text{RGB}_{x+1,y}, \text{Horizontal } \Delta\text{Hue}_{x+1,y}))} \quad (48)$$

Depending on the filter pattern implemented by the camera, there may be consistencies in terms of the proportional change between the boundary pixels for the three colours, but cannot be guaranteed. If Moiré patterns are present, the difference between the three colours can typically fluctuate, thus allowing the removal of the candidate.

To improve the consistency of the surface around the boundary, the horizontal neighbours can be included in the analysis. The significance of finding a surface, and not just a simple line, lies in the applicability to the local map as thin strips may not be registered by the range finders, thus introduces an unnecessary processing load in attempting to find a fit on the map. By noting that the boundary is part of a larger surface, it can easily apply the features with more confidence and relevance.

Since the relationship between the number of pixels shown and the size of the actual object is dependant on the distance, it is difficult to effectively determine the appropriate number of pixels required for the granularity of the local map dynamically without introducing dependencies back to the local map. Instead, a fixed number of pixels is used, which is derived based on an object at a distance of 1m, the size of the object being 1 cm, and the viewing angle of the camera being approximately 40 degrees, which equates to approximately 5 pixels when using a capture resolution of 320 x 240 pixels. This value means that if the object that is 40 cm away is viewed at, the object must have the same intensity value for 4 mm on either side, while if the object is 5m away, the object need to be consistent for 5 cm.

The fluctuations in the intensity that are seen for the extra neighbours should remain small, but must allow for small changes such that the noise and subtle lighting changes still allow the features to be registered. To account for the various sources of noise, the edge scores evaluated earlier can be used as it suppresses the effect of lighting changes using the hue and already combines the RGB intensities. Since it is the consistency in the intensity that is required, the maximum score of the

### 10.3.1 Feature selection

two colour scales will be used. By setting the threshold to only allow the features with a low score, the neighbouring pixels of the boundary will remain consistent. When combining the scores of the four horizontal transitions, two basic strategies can be used, which are the use of the maximum difference and the average transition score. Depending on how flexible the thickness of the consistently coloured surface must be, the algorithm can be interchanged.

To differentiate the difference between an edge and noise, the region above and below the boundary is observed on top of the horizontal neighbours. This process allows the continuity of the edge to be monitored, such that the end points of the edge or gaps in the line can also be observed. Due to the slope of the edge, interpolation, and changes in the object structure, the presence of the edge may not always appear directly or diagonally above the current edge. In this case, the boundary can be marked as the end point or the search can be extended until another boundary is found or the distance becomes too large.

Although this approach allows for some level of continuity between the boundaries, it is heavily dependant on the threshold conditions and can behave quite differently depending on the texture patterns. If there is a slanting of the boundary, the interpolated intensities will cause some blending in the horizontal direction. However, this means the vertical transitions should not be affected for a vertical line if the difference is measured at the boundary that is slightly away from the horizontal transition. Instead of rigorously making sure the gap in the intensity is indeed from two surfaces meeting, the check is only carried out to four vertical boundaries around the horizontal transition. This results in a faster processing of the feature candidates, but it can also mean the end points to the edge will not be detected. Since the exact height of the edges are not important on the local map, as it only maintains a 2D model, this is not a significant issue, other than when determining the continuity, which will be described later.

Since there are two surfaces to consider, it is not crucial for both sides to remain consistent. This means the better score of the two sides of the boundary can be used. This concept also applies to the horizontal neighbours, which selects the better score of the two sides. When comparing the transitions above and below, similar consideration must be made as the horizontal neighbours, but with slight differences in that the minimum value can also be considered here due to the proximity of the region to the boundary and also the slightly different purpose it has, as it can encounter the end of the vertical line. Figure 10.10 illustrates the process of determining the consistency in the vertical direction, which uses the minimum difference between the two pixels. The letters A to F represent the intensities at the corresponding pixels.

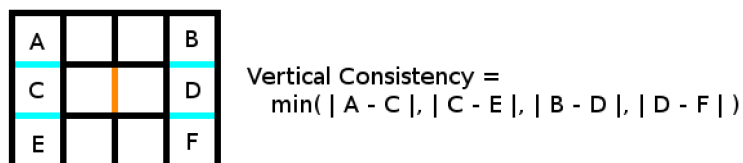


Figure 10.10: Vertical consistency in the edge surfaces. The central boundary indicates the edge of interest and the outer boundaries represent the vertical boundaries that are considered.



### 10.3.1 Feature selection

In terms of the processing requirement for this task, it should become apparent that the comparison with the pixels above involves a vertical edge, which is not included in earlier calculations. This means that the vertical edges between the pixels must also be calculated. If the selection criteria for a feature can be partially completed using only the transition score and the neighbouring consistencies, some of the candidates can be eliminated to improve the efficiency. Otherwise, the iteration can be carried out vertically first as it will reduce the buffer size that is required, but should consider the offsets too.

The use of the three scores can be done incrementally, which can allow early elimination of candidates, or the scores can be combined for a single evaluation process. This can use a threshold value by setting a required score or by ranking the candidates and selecting the better scoring candidates. The combination of the three values can be approached in a similar way to the two colour scales, but the emphasis must be placed on the presence of the boundary over the other two. This can be done by eliminating some of the candidates early using just the edge transition score or by applying a greater weight to the surface boundary. The use of a multiplication does not allow the distinction between the different sources, thus cannot be used reliably here.

Using just the boundary score yields a similar reduction process to the temporal filter, except those marked from the edges of the previous frame. Since the number of candidates has already been reduced, the boundary score is used in conjunction with the consistency scores in a single process to assign an overall score to the remaining candidates, as shown in figure 10.11. The colour coding shows which pixels the different scores are derived from, where the red represents the consistency weight, yellow represents the boundary score, green and cyan represents the horizontal and vertical consistency scores respectively. The score allows the candidates to be ranked such that further elimination can be made based on the processing capabilities instead of the characteristics of the current view.

The derivation of the feature score is quite a lengthy process compared to a more naïve edge strength approaches, thus is important to make use of buffers to re-use some of the computations. It is also possible to modify the threshold of the temporal filter to reduce the initial number of candidates if the processing time is too large. Other approaches are also introduced to reduce the candidates, such as only evaluating the score to newly introduced features. It is important to apply these other candidate reduction processes beforehand, as it can eliminate the candidates much more efficiently than using the score. Other strategies that are incorporated are described later on.

Although the evaluation of the score depends on many pixels surrounding the boundary, the attributes that characterise the feature does not have to contain as much information due to the changes that may occur between the frames. The important attributes that can currently be defined are the coordinate points, the two surface intensities bordering the boundary, and the score. These attributes alone will not always be able to uniquely distinguish the feature in between frames, thus additional considerations will be made.



### 10.3.1 Feature selection

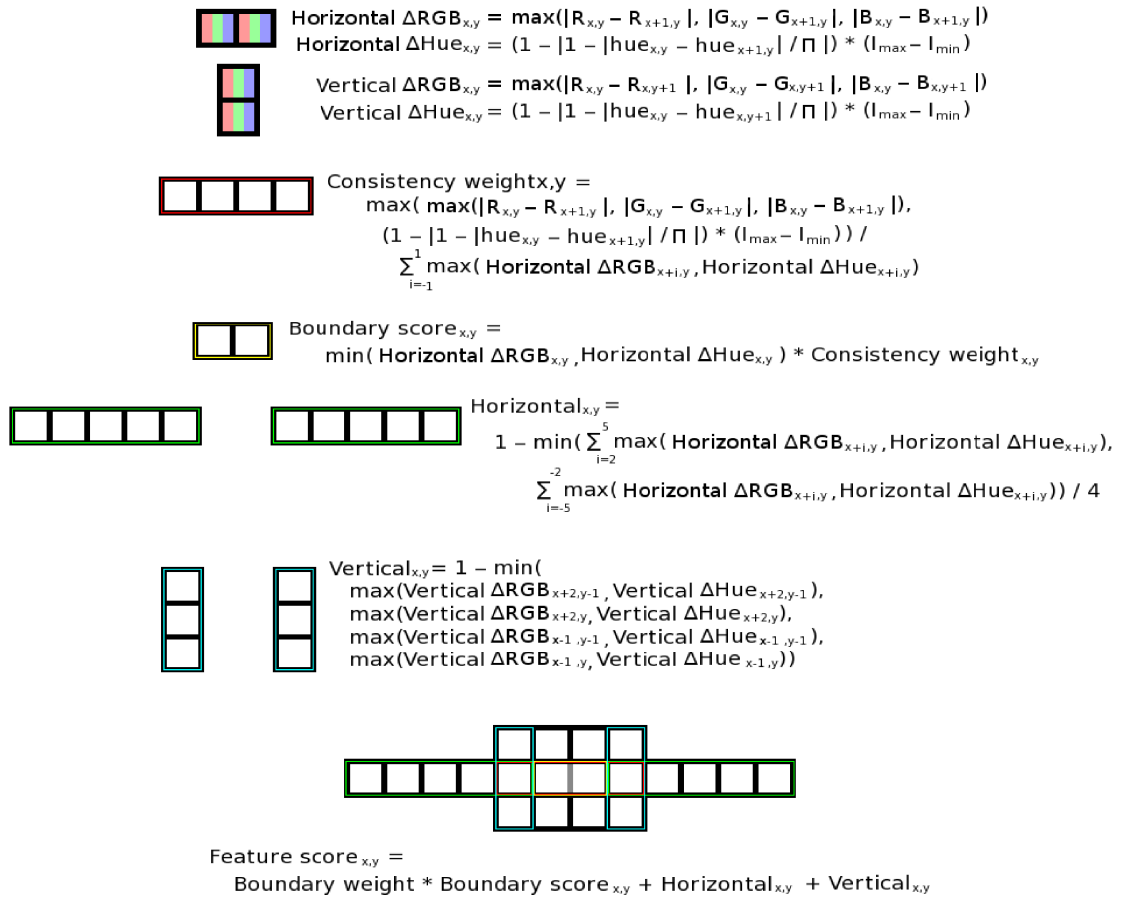


Figure 10.11: Derivation of the feature candidate scores.

Top two illustrations are used to determine the difference in the colours. The red region is used to determine the proportionality of the edge strength, the yellow region is used for the boundary strength, the green region is used to determine the consistency of the surfaces in the horizontal direction, while the cyan region determines the consistency of the surfaces in the vertical direction.

### 10.3.2 Surface feature

When using an edge based feature detection to identify interesting regions of the image, it is important to note that the transition in the intensity values is often derived from one object obscuring another in the background. This means that when the feature is observed from a different perspective, the texture of the background object can change dramatically and modify the appearance of the feature region. This can cause a lowering of the correlation score and result in an incorrect match between the stored feature and the current view.

As hinted earlier, the scores of the left and right side of the boundary can be treated independently, such that the changes to one side of the boundary do not worsen the scores. By splitting the feature into two components, it can potentially double the number of candidates, unless the uninteresting features are removed quickly and efficiently. Another issue this causes is the reduced number of attributes

### 10.3.2 Surface feature

that are assigned to the feature, which can contribute to the ambiguity in distinguishing the feature.

Due to occlusion, it is more desirable to track the motion of the surface in the foreground, as the surface boundary for the background may not actually exist. Although the distinction is quite difficult to determine, by maintaining the foreground feature, it can eventually cover the background such that the background feature will be lost. Until this happens, the boundary of the background will appear to be attached to the foreground object, thus will be treated as if it is a corner between the two surfaces. This ambiguity can be removed by combining other sources of information, such as the depth queues from the local map.

An alternative approach uses the correlation score on the better of the two sides, but maintains both sides in the single feature. This procedure can maintain the tracking of the foreground and background surfaces without having to duplicate the feature, as the motion of both sides will remain the same regardless. By monitoring for a large fluctuation in the score in one side, the side can be disregarded from further calculation. An example of this is if a boundary that exists on the background becomes visible, the foreground feature can now disregard the correlation score on that side while a new feature can be tracked separately for the background feature.

### 10.3.3 Density

The large number of potential features that must be validated and tracked means the maintenance aspect can consume a significant amount of resource. Depending on the feature candidate's scoring process, many features can be introduced from a single boundary as multiple snapshots are taken of a very similar view of the same location. Since there is limited number of pixels, it is possible to find the upper limit on the number of visible features to limit the overlap of features. Even after limiting the maximum number of features, the image's resolution may allow for too many features to be processed in real time.

The typical strategy for tackling this is to find the features that are more constrained, such as corners or a scale and transform invariant features (Wang & Brady, 1995). Although these are quite useful in identifying distinctive features, the focus of the feature detection here is to determine the structure of the environment and improve the precision of the local map. Since the features are for short term use only, they do not require the robustness and is required to focus on successfully tracing the surface boundaries.

The majority of intensity and edge based feature detection algorithms form highly dense clusters, as the objects that are in the view are of a reasonable size and typically have a consistent textured surface. Since the features within the cluster are often from the same object, their characteristics will be repeated many times, with the exception of the slight differences in the coordinate point of where it was observed. By compressing these features into a simpler representation, the number of features that are maintained can be drastically reduced.

Using a detailed clustering analysis to identify the ideal reductions can potentially require a lengthy processing of the current view, the features, the state of the local map, as well as historical information on the trends of the features. To save on the

### 10.3.3 Density

processing time, a simple proximity based filter is used to eliminate some of the features that are close together. Several attributes for the filter are considered to determine the appropriate density and distribution of the features.

The simplest filter to be implemented is a sequential square shaped mask which eliminates all but one feature within the mask. This allows for a significant number of features to be removed, but since the features are based on horizontal differences, a sequence of horizontal features will mean there should be two distinct vertical lines. Eliminating these closely located lines can be problematic if their information is permanently lost, thus the width of the mask must be shrunk down.

The size of this mask has a direct impact on the number of features that will be kept, but this depends on the amount of approximations that can be made about the scene structure. The reduction in the candidates should not lose significant amounts of information about the structure of the environment, as ideally, the lost information should be able to be reconstructed without diverging too far from the original appearance. If the approximate object size is known, this can be applied much like the procedure used in the horizontal neighbours detection. Using a rough measure of height, the size of the mask can be constrained.

The selection of the features to maintain can be done based on positional, randomised, or score based algorithms, which all have different side effects that require consideration. When using a position based approach, the mask must be applied with an appropriate offset to the image, as the features and the expected position in the mask may not be aligned properly. This approach also requires the positioning of each of the masks to be considered, as the arrangement causes different number of features to be eliminated. It is also possible to scan for a feature within the mask, which is set at a particular position, thus allowing a more consistent number of features.

Using a randomised algorithm to select the feature can be effective in situations where the scene characteristics are unknown, but can often result in a non-optimal arrangement of features being selected. Since some of the scene characteristics can be anticipated, such as the presence of vertical lines, a more predictive approach can be used instead.

The score based approach requires some form of ranking to be applied. Since evaluating the edge score and ordering the features can consume a significant amount of time, the process must consider a simpler attribute to base the score on. This could be based on the amount of change that was observed from the temporal filter to eliminate the candidates that have similar intensities to the surroundings. However, this process is integrated with the threshold at the temporal filter, thus does not need a separate parse of the image.

The current mask is based on the idea that one of the key element to be removed should be the repeated edges caused by interpolation from the aliasing, and that the re-construction of the object boundaries requires a reasonable number of features in line to form the boundary shape. This aspect will be covered in chapter 11.

The mask, as shown in figure 10.12, is applied in two distinct stages, where one involves the elimination of the feature candidates before the scores are evaluated, and the other after. The highlighted regions in the left image are removed first, while the

### 10.3.3 Density

second phase on the right filters out the lower scored pixels. The first filter allows halving of the feature candidates while allowing consistent coverage of vertical lines. The toggling of the mask between the lines allows more of the non-perfectly vertical lines to be observed, as they may intersect the horizontal boundary of the masks. The mask can be problematic if there is a diagonal line in the scene which falls perfectly in the blind region, but since the primary focus is on the detection of vertical or near vertical lines, this is not applicable here.

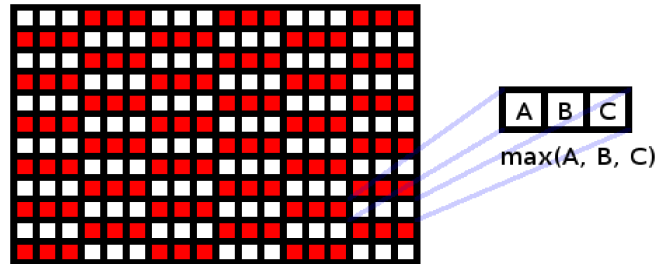


Figure 10.12: Feature density reduction mask.

The highlighted cells indicate the cells that are initially removed for consideration. The group of three pixels are then compared to only maintain the best out of the three.

Note that the use of the filter has a similar performance benefit to halving the capture resolution, but selectively eliminating the unnecessary pixels to maintain the same level of precision for the features that do remain. The small spacing also allows scope for the features to be joined together by assuming continuity, which becomes more difficult with increased spacing between the gaps.

The second mask is applied to the group of three pixels by comparing the scores and discarding all but the highest scored feature. This allows the reduction of consecutive horizontal edges being formed, except in between the different rows. Although the consistency scores that are evaluated allow the removal of some of these features, this filter allows for extra reduction of these redundant features.

Although it is ineffective to implement with the current set of filters, it is possible to use a shifting window to move horizontally to determine the highest scored feature. This will avoid the misalignment issues between the objects and the mask, but also re-introduces the double edge problem if it does not explicitly exclude adjacent features being selected from different window positions.

Two more strategies are introduced to reduce the number of feature candidates, which consider the current group of features and the rate of encountering a new feature. The first approach distinguishes whether the region within the image is an existing feature or a newly introduced feature. As noted earlier, once the feature has been found, it does not require a complex analysis to maintain the tracking. This means the regions that overlap with the existing features can also be removed from the list of feature candidates. This can also include the adjacent positions to the existing features, as they are likely to be a part of the same feature. Since this process requires the current position of the features, the new feature detection process should occur after the tracking of current features has been completed. One extra consideration to make when using this and the filter used above is when the foreground feature and background feature separates. Since the newly created feature

### 10.3.3 Density

for the background will be away from the foreground feature, the elimination of one side is required to monitor a significant change in the appearance instead of relying on the creation of a new feature.

The second strategy involves using the domain knowledge about the speed of the robot and the change of camera positions, which contribute to how frequently new features can be introduced into the view. An important note to consider is that the detection of these features is not critical to the operation of the robot, as it is simply an enhancement to the existing map. With this in mind, a delay between the new feature detection processes can be introduced. The waiting period can be based on time, the actual change in the pose of the camera from the localisation module and the positions of the servo motor, or even by simply counting the number of pixels that are observed to be different through the temporal filter.

Since the frame rate remains reasonably consistent, delaying by a set time can be implemented by simply using a counter on the number of frames that are captured. However, this does not perform as efficiently as the other approaches, as there is no guarantee that the image has changed. Using the motions of the robot and the camera, it is possible to anticipate approximately how much of the scene is introduced into the view, but it is unable to cater for dynamic objects, as well as not being able to predict the structure of the scene behind a foreground object. Instead of combining the two approaches, the third approach of detecting a large change in the number of pixels that are found to be different from the temporal filter is used to trigger the feature detection process.

The threshold value for this must also consider the density filters and the existing features that will reduce the number of candidates, since specifying an upper limit to the features will allow the algorithm to perform at a consistent rate. With the current density masks, the maximum number of features that are ever possible is approximately  $\text{width} * \text{height} / (3 * 2)$ , which equates to 12,800 for a capture resolution of 320 x 240, which is quite large. Depending on how the features will be used, the required density can differ greatly. In the current implementation, the typical scene structure can be taken into account to predict the expected number of vertical lines in a single view. If there are 10 or so vertical lines within a the view with an average height of half the height, there can be approximately 600 features in the scene. Based on this value, a threshold value of 512 is used to trigger the feature detection process if the number of features being maintained drops below this point.

With the scores determined for the feature candidates, they can be sorted so those with a higher score can be maintained. Instead of implementing a generic sorting algorithm, it is important to note that the candidates do not have to be in a fully sorted sequence, as long as the resulting number of features is within a reasonable range. Since the actual upper limit for the number of features is too high to be managed in real time, a limit is defined to be twice the threshold used above. With this in mind, the sorting of the candidates only requires a simple split such that the number of candidates plus the current number of features is below 1024 and above 512. Note that these values should be modified depending on the type of environment the robot encounters, as inappropriate values can trigger too frequent feature detection or miss out on important views of the scene.

To determine the top candidates, algorithms similar to quick sort or bucket sort

### 10.3.3 Density

can be used, but focusing on counting the number of elements that are in the group. It is possible to keep track of the maximum and minimum candidate scores to assist in defining the pivot value, or the bucket size, but it is also possible to simply make use of the theoretical upper and lower bounds. Algorithm 10.1 illustrates the process using a bucket size of powers of  $2^4$ , which has been selected based on the level of precision of the various components of the feature score. Note that by controlling the two weights and the level of precision used for the hue value, it is possible to map the scores onto a fixed point decimal number representation like the one used in chapter 9, which can improve the efficiency of the bucket algorithm.

```
function FilterCandidate(features, candidates, upper, lower):
    set factor = 2^4
    set bucket[factor]
    set precision = 1 / factor
    while feature.size < lower && candidates.size > 0:
        for i in 0 to candidates.size:
            set index = ⌊ candidates[i].score / precision ⌋ %
                factor
            add candidates[i] to bucket[index]
        clear candidates
        for i in 0 to factor:
            if feature.size + bucket[i].size > upper:
                set candidates = bucket[i]
                clear bucket
                set precision = precision / factor
                break
            else:
                for j in 0 to bucket[i].size:
                    add bucket[i][j] to features
    return features
```

Algorithm 10.1: Candidate elimination from counter based threshold.

Although the filter above can assume an upper limit to the number of features, the low scoring features that are most likely erroneous are still maintained if the total number of features within the scene is small. To allow the elimination of a badly matched feature is by slowly decrementing the feature's score by using the inverse of the correlation score. Once the score is reduced to below zero, the feature can be eliminated.

### 10.3.4 Tracking

To determine the structural information of the scene from the features, their motions must be monitored against the changes in the perspective to allow triangulation of the pose of the feature. The tracking of the same object also assists in disambiguating other attributes by allowing the observations to converge to a constant value. By using the unique attributes that were determined during the feature identification process, the same object can be found in the subsequent frame, but will likely have undergone small amount of translation and transformation, thus a correlation score must be evaluated to find the most likely match.

Since the distance to the object is reasonably large, the amount of motion that can occur in between the frames is quite limited. This is true for when the camera translates, but can be problematic if a fast rotation occurs. By limiting the rotation of

### 10.3.4 Tracking

the servo motors to delayed increments, it is possible to limit the range of motion in between the frames. The operational speed of the robot is currently set to be quite slow to improve the effectiveness of the localisation algorithm, thus the search space for the features can remain quite small.

One of the constraints that is placed on the features is that they are used to detect vertical lines, but at the same time, it does not distinguish the vertical intensity transitions. This means any features that are not along the mid-height level will lose its vertical pose as the camera moves around. If tracking of these features are attempted without any vertical intensity changes present in the feature, the continuous surface of the objects will allow the features to slide up and down with little changes to the correlation score. Since this would eventually result in features overlapping with each other, many of the features will quickly disappear before reliable information about the feature's motion can be derived.

To prevent the features from sliding up and down, it may be possible to anticipate some of the vertical motions by using the current position of the feature within the view and the pose changes of the robot. However, since the pose changes may not be available, the constraint may have to be derived from historical data.

An alternate approach is to simply not worry about any elevation based information and only track the horizontal motion of the feature. This places an extra emphasis on tracking the surface boundary, as the tracking will no longer be based on a specific point along the boundary, but the boundary itself. This means new features can also appear above or below existing features as the robot moves towards or away from the boundary. This also means that the errors from slight bumps and misalignment of the camera will be reduced.

When implementing this approach, the image plane should be parallel to the vertical lines, as any difference in the pitch will result in the features moving in an arc across the view when the camera is rotated. If the camera is tilted, the transformation of the image to a perpendicular view with respect to the ground can result in unnecessary interpolation with vertical neighbours, thus should be avoided when possible.

Due to the staggered motion of the servo motor and the slow rotational speed of the robot, the maximum horizontal motion between the frames was observed to be approximately 8 pixels with a horizontal resolution of 320 pixels, which is used to limit the search space for each of the features. Since the motions of the features are related to the objects, the motion of one feature can be used to guide the tracking of the other features. If a rotation occurs, the horizontal motions of all the features should be reasonably consistent, while a translation should result in groups of features, especially those along the same horizontal positions within the image, moving in a similar manner. If the features can be arranged in a way such that they can be accessed according to their horizontal positions, such as a two-dimensional array, it is possible to track the displacement of one of the features along that vertical line and use that position as the starting position of the search.

Since the commands sent to the motors are carried out over a period of time, the direction of the feature motion can also be anticipated using this information. As with the motion prediction, the initial search location can be modified to the entire set of features, as their motions do not differ greatly to warrant an individual prediction,

### 10.3.4 Tracking

such as by using historical information.

Although there are multitudes of algorithms that can be included to try and improve the efficiency of the tracking, the overheads that are introduced can be far greater than the benefit they provide. Since the attributes that are stored for the feature is quite simple, the processing time for each feature is not a significant load. Given that the search space only consists of 17 positions, the main contributor to the processing load is the number of the features and not the processing of the individual features. With this in mind, the predictions to improve the search speed for the feature location has not been included in the current implementation.

The two surface intensity attributes that are stored for the feature can be used to determine the correlation score based on the distance measure of the intensities. However, since only one side is required to match against the feature, using a single pixel does not yield a reliable feature tracker. To counter this, three different strategies are incorporated to assist in the disambiguation process, such that the feature tracker allows for the same surface boundary to be monitored.

The first strategy analyses the surrounding intensities of the feature, such that the neighbours are checked in a similar fashion to the feature detection process. Instead of the irregular shape it observed previously, the comparison is conducted on a small square region to simplify the computation. The score that is derived is a measure of consistency based on the pixel intensity of either side, thus the difference in the intensity can be re-used as the surface region being observed shifts. Using a region of size 3 by 1 pixels and starting the scan from the adjacent position to the edge, the boundary interpolation can be eliminated while allowing the consistency of the surface to show. Figure 10.13 illustrates the region being compared with respect to the position of the feature, where the three pixels in the red rectangle is compared against the feature's colour and added together.

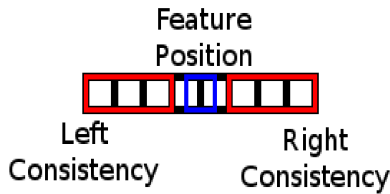


Figure 10.13: Feature tracking region of comparison for consistency in surface.

The blue square represents the location of the feature, while the red rectangles portray the surfaces that make up the boundary.

Note that this may include pixels that are still in transition, thus may require weighted adjustments to the consistency value to make the contribution less significant. The actual score can be evaluated using the following formula.

$$\text{Consistency}_{L_{x,y}} = 1 - (\text{Boundary weight} * (| \text{IFeature}_{x,y,\text{left}} - I_{x-1,y} | + ( | \text{IFeature}_{x,y,\text{left}} - I_{x-3,y} | + | \text{IFeature}_{x,y,\text{left}} - I_{x-2,y} | ))) / (2 + \text{Boundary weight}) \quad (49)$$

$$\text{Consistency}_{R_{x,y}} = 1 - (\text{Boundary weight} * (| \text{IFeature}_{x,y,\text{right}} - I_{x+2,y} | + ( | \text{IFeature}_{x,y,\text{right}} - I_{x+3,y} | + | \text{IFeature}_{x,y,\text{right}} - I_{x+4,y} | ))) / (2 + \text{Boundary weight}) \quad (50)$$

$$\text{Consistency}_{x,y} = \max(\text{Consistency}_{L_{x,y}}, \text{Consistency}_{R_{x,y}}) \quad (51)$$

Using just the above approach, it should be apparent that just using the consistency measure does not allow the actual boundary region to be accounted for.



### 10.3.4 Tracking

By storing the intensities at the boundary, the transition characteristics remains fixed between the specific foreground and background intensities. Instead of using the original intensities, the total strength of the boundary can be determined from the current view by accumulating the average intensities of the two regions during the above process. The derivation of the difference does not need to be as rigorous as the feature selection process, as there is no guarantee that the boundary still exists as the view may have changed.

$$\text{Transition}_{x,y} = | (I_{x+2,y} + I_{x+3,y} + I_{x+4,y}) - (I_{x-3,y} + I_{x-2,y} + I_{x-1,y}) | / 3 \quad (52)$$

Since the pixels used to measure the transition do not actually include the pixels at the boundary, there is no guarantee that the proposed position contains a change in the intensity or the same transition as the surrounding pixels. To encourage the surface boundary existing in the middle of the observed region, this can be included in the correlation score.

$$\text{Boundary}_{x,y} = | I_{x+1,y} - I_{x,y} | \quad (53)$$

$$\text{Correlation}_{x,y} = \text{Boundary}_{x,y} + \text{Transition}_{x,y} + \text{Consistency}_{x,y} \quad (54)$$

Even with the proposed technique, the precise location of the surface boundary cannot be determined. To derive a more accurate position of the feature, the tracking algorithm must maintain historical camera pose and feature locations such that the actual position can slowly converge to the desired value.

The last of the three strategies that is introduced is the notion of continual motion, which is determined from the previous feature motions. In the localisation algorithm, this information is used to specify the starting location for the search. The lack of reduction in the search space means the tracking will still be successful even if the prediction is incorrect. One of the characteristics that make the early termination attractive is if the scores that are evaluated is quite different. However, this means the surface should be significantly different for the scores to fluctuate to reach the threshold quickly.

Instead of designing a search sequence altering algorithm, the search space itself can be reduced to improve the efficiency based on the previous feature motion. Since the majority of the motions that are detected will be continued on from the previous motion, a smaller window can be placed around the predicted area. This can potentially mean that the correct feature position will not be found if the motion suddenly changes, thus relies on the correlation score to indicate that the feature was not found.

The size and placement of the window depends on location of the feature, as well as the rate of change, thus maintaining all these attributes for each of the feature can amount to a reasonable footprint size. Instead, the minimum and maximum motions of all the feature motions are maintained as one and used for all of the features. Since this value can change between frames due to changes to the motion pattern, an extra 4 pixels are added to the magnitude of the two limiting values. Since the maximum motion has been defined, the two limiting values can be trimmed to this value. Note that ideally, the maximum motion of the feature should not be reached since the feature tracking should be carried out while the robot is in motion and the servo motor being stationary to track the feature from translation and not rotation.

The evaluation of the scores introduced above is based on a single dimensional

### 10.3.4 Tracking

intensity measure, thus must be modified to account for the three different colours that are available. For the correlation score, the value must indicate how close the values are, thus the maximum difference should be used. As for the boundary score, it is important to note the presence of the difference, thus the maximum difference can be used as well. Note that the hue scale is not used for this analysis, as it deals with the bordering region of two intensity changes, which can cause large fluctuation in the hue.

The last issue to consider with regards to the feature tracking is the viewing area, which is dependant on the radial warping from the camera, as well as the visibility of the feature in the subsequent frame. Since the cropping from the radial warping is currently set at an arbitrary amount based on the apparent interpolation, the focus is placed on the visibility aspect. Currently, the feature selection requires 12 pixels horizontally and 3 pixels vertically for each of the features, while the tracking requires 4 pixels on either side of the feature and 8 pixels in each direction as potential locations of the feature. This means there should be at least 12 pixels of buffer region to the side of the frame and a single row at the top and bottom where the feature should not be maintained. Using these values, a border region of 16 pixels is placed on the sides, while the top and bottom have been trimmed by 8 pixels to reduce the effect of warping and discourage features that are too high for the robot to be included.

## 10.4 Map enhancement

The tracking of the features allows the inter-frame behaviour of the particular boundary to be monitored with respect to the robot and camera's pose, thus allowing the derivation of the boundary position through triangulation. Using the stored intensities of the surfaces, it is also possible to assign textures to the surfaces for a more complete re-construction of the environment to assist in the navigational tasks for the robot (Seitz & Dyer, 1997).

### 10.4.1 Depth from motion

As the local map focuses on identifying the locations of the objects in the scene, the features that are tracked can assist in enhancing the location of the boundary regions, which will appear as corners or possibly as features on a flat surface. This process can be achieved in two ways, where one involves the build up of the boundary location before being applied to the local map and the other involving continuous superimposition with the local map.

Many existing structure from motion approaches are based on a simple triangulation algorithm of disambiguating the location of the feature, which can be achieved through two different positions of the feature and the corresponding camera pose (Matthies et al., 1988). The technique is similar to those used in parallax or stereo depth mapping techniques (Bleyer & Gelautz, 2004; Hemayed et al., 1997 (a); Iocchi & Konolige, 1998; Irani & Anandan, 1996; Kumar et al., 1994; Murray & Jennings, 1997; Rosselot & Hall, 2004), which require precise camera poses and the feature locations. If other constraints are available, it is also possible to derive this with increased views from different perspectives (Quan & Kanade, 1997). Figure

### 10.4.1 Depth from motion

10.14 illustrates the various components involved in the triangulation process for two known camera poses.

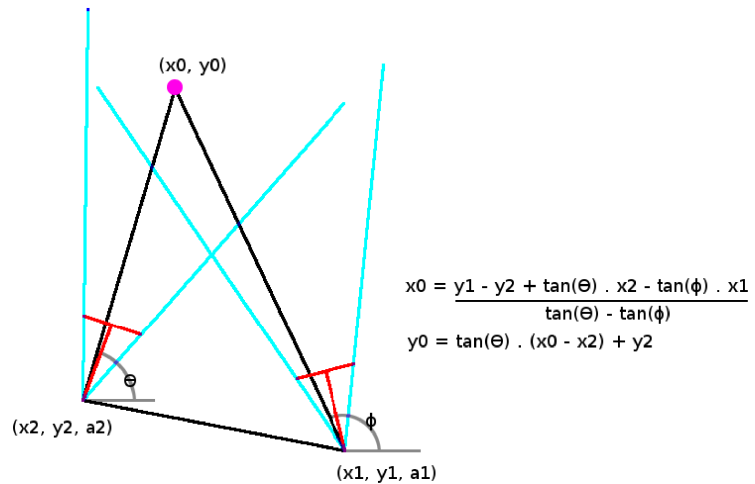


Figure 10.14: Depth detection from triangulation.

The circle represents the object of interest, the light lines represent the limits of the viewing angle, and the T shaped lines show the projected image and the perpendicular line at the center. This corresponds to the orientation of the pose.  $\theta$  and  $\phi$  are the orientation of the object of interest in the projected image with respect to the global coordinates.

Since only one camera is used in this set-up, it is vital to minimise the pose errors for the cameras. Coupled with the precision errors in the tracking and the increased ambiguity of the feature's location with increased distance, the depth value that is determined from the tracking will often be inaccurate if only two poses are used. Since a large number of poses and the feature positions can be made available through continual tracking of the feature, it is possible to combine the multiple depth evaluations to converge the pose to a more accurate value.

When the multiple measurements are conducted, a similar consideration to the carving can be applied, where the change in the orientation between the tracking is used to weight how much of an effect it has on the local map. Since every pair of feature position with respect to the camera pose can be combined for the triangulation, the number of pairs increases at the rate of  $(N^2 - N) / 2$ , where  $N$  is the number of tracking. Given the high frame rate, this can quickly become an issue, especially because there is no upper bound defined except when the feature is discarded.

Since combining consecutive tracking does not allow for the orientations to differ greatly, it would be more ideal to combine those that are taken with some time apart. Two different strategies are considered to allow the limiting of the memory and processing requirements while encouraging the use of pairs that are taken at some distance apart. The first approach involves maintaining a running average, while the second maintains a selection of feature positions to keep the outlier features.

The maintenance of the running average involves using the initial feature position as the constant reference point and continuously evaluating the feature pose. By

#### 10.4.1 Depth from motion

maintaining a counter, it is possible to accumulate the values, which can be averaged at any time. Note that in certain cases, the feature will remain stationary within the view, such as when the object is very distant, the rotation of the camera counteracts the motion of the robot, or when no motion occurs. Since this can bias the pose from the lack of the change in the perspective, these repetitive tracking can simply be discarded. The approach also depends greatly on the accuracy of the reference position, as this can cause a misalignment of the object. Since the accumulation of the pose is required for each pose of interest, the number of calculations can become an issue as the effect of additional poses decreases over time.

The second approach attempts to reduce the issues from the above by reducing the number of pose evaluations to only those near the initial position of the feature and those just before the feature is lost. This can greatly reduce the number of evaluations, especially if the positions that are maintained are kept small and those that are stored near the start is separated from those just before the feature is lost. This approach assumes that the motion of the camera does not return to the same position when it is lost, as the orientation will be quite similar. It can also discard the majority of the tracking information in between the start and end of the feature being tracked, which can provide many different orientation of the same object.

Instead of always selecting the first and last few positions, it is also possible to randomly select the candidates or selectively decide which positions to use based on the camera pose. Depending on the availability of memory, the complete trace of the different positions can be stored for each of the features to be analysed in detail. Since this can result in an enormous amount of wasted memory, a dynamic selection approach is applied.

For the random selection, the frame number and the number of positions to be maintained can be used to determine a probability of it replacing one of the current entries. The replacement of the existing store of positions can be done randomly or sequentially, which must consider the special case where the number of positions is smaller than the storage size.

The selective approach is slightly more processing intensive, but can be designed to distribute the feature positions by eliminating those that are closely located to other entries. By summing the difference in the orientation between the other entries, the one carrying the minimum distance can be discarded. It is also possible to simplify this by maintaining the sorted positions and only using the distance to the adjacent orientations.

The result of the depth from motion approach is shown in figure 10.15, where the surface boundaries are superimposed over the local map. The reliability of the range finders meant the two approaches could not be compared accurately, but the selective algorithm did not show any noticeable increase in the processing time, thus is used in the current implementation.

Note that although the current implementation does not consider the vertical edges, similar strategies can be applied to determine the height of objects by observing the upper and lower most features of those in a line, or by introducing the vertical edges in future implementations.

By using a single image, it does not allow the identification of the depth. By

### 10.4.1 Depth from motion

combining the camera orientation with the local map, it is possible to correlate the direction of the feature with the occupancy on the map to identify the object's location.

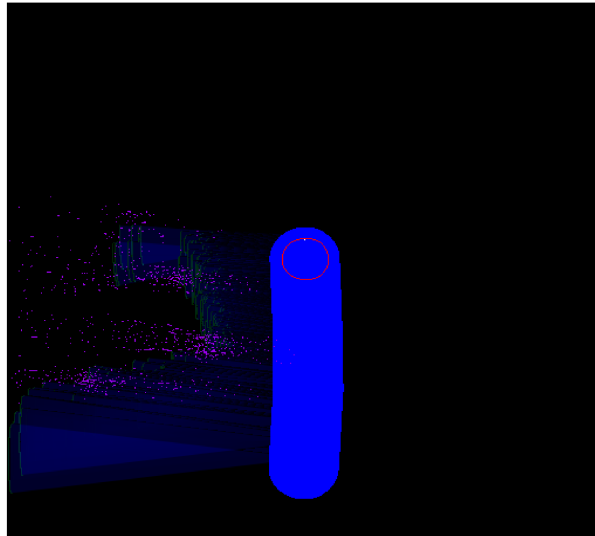


Figure 10.15: Surface boundary positions superimposed over the local map.

The blue is used to represent vacancy, green for occupancy, and the red circle is the location of the robot body. The purple dots scattered on the left hand side are locations of the features when they were mapped onto the map.

When representing the visible feature as a line or a cone from the camera, it is possible to trace its trajectory onto the local map. This can then perform a similar occupancy accumulation like the carving, but based on a pure accumulative value, since no limit to the distance is used. This process was required for the range finders, because it was not possible to verify that the tracking of the same object in between the scans took place. The map itself was used instead to maintain the possible locations of the objects.

Instead of maintaining another attribute at the local map, it is possible to simply add to the occupancy value of the cells that intersect or are close to the possible location of the features. The scores can be modified based on the distance to the line, which eventually terminates once it encounters a cell that is strongly marked as being occupied or gradually distributes the weights as it extends out further, such as by using a formula like below.

$$\text{Weight} = \text{Occupancy} / \text{Distance} \quad (55)$$

The individual carving of the lines can lead to increased processing requirements, and does not improve upon the previous approach, as the tracking of the feature is not well utilised and is dependant on the completeness and accuracy of the local map. With this in mind, the tracking approach is used, which calculates the pose of the feature until they are lost before being applied to the local map.

One of the possible issues with this approach is in the latency between the localisation algorithm and the observations made by the camera. Although the delay is small, the errors in the robot's pose can result in incorrect feature poses being

#### 10.4.1 Depth from motion

calculated. Since the localisation algorithm itself introduces a slight delay to smooth the motion, an alternate method of synchronisation is required. In the current implementation, a constant offset has been derived using the same technique as the localisation cameras. Unlike the two manually colour adjusted cameras, the automatic colour adjustment on the camera caused some difficulties when the light being introduced was too different. Instead, the cameras were both directed to a surface where a small point was illuminated by a red LED. This test showed that the horizontally viewing camera was delayed by 0 to 3 frames depending on the amount of ambient light present. Since there is a delay in the response to the motion due to the smoothing carried out within the localisation algorithm, the majority of the motion is delayed by 1 frame. Using these information, the motion from the localisation is not delayed any further, as the lighting condition of the environment is often reasonable to keep the delay in the horizontal viewing camera low. There is scope in the future to dynamically adjust this value depending on the current settings for the colour adjustments done by the camera.

The effect of adding the feature pose to the local map is yet to be fully explored, but can range from defining a more precise corner of objects to joining these feature points for a surface reconstruction, which can make use of the existing occupancy values to disambiguate the scene structure. The combination of the depth structure that is derived with the arrangement of the intensity in the captured image can also lead to unique identification of the object being observed (Lowe, 1987).

#### 10.4.2 Texture mapping

An obvious application of the visual sensor is to map the textures of the objects onto the surfaces of the local map (Debevec, 1996; Heckbert, 1986). As mentioned earlier, since the flat surface is not tracked as a feature, the intensity found on the side of the feature can be used as a simplified representation of the surface texture. One important issue to note is that the local map currently does not store any elevation related information. This means many of the intensities will conflict with each other as multiple surfaces at different heights can be observed. To counter this, the map must filter the features such that only those that correspond to a surface that is found on the local map to be used. This concept should also be applied to the depth map, but will be discussed in more detail in chapter 11.

The height of the object that should be detected is dependant on whether it will collide with the robot, thus the camera is unable to see the whole object if it is too close. Instead of attempting to derive the height, the purpose of this texture mapping is observed, which is to assign an extra attribute to the cells such that the correlation can be made more accurately. This means that as long as the intensity texture that is assigned is consistently observable for that surface, it does not matter at what elevation the intensity is extracted from.

Since the height of the observed objects change if they are not viewed parallel to the motion of the camera, the features along the mid height level is chosen for the texture extraction process. Note that since the density mask trims half of the features in one row, the two middle rows are combined to represent the whole row. The features can then be used as the bounds for applying the intensity measure, which can make use of the adjacent intensities of the feature itself, or determined through

## 10.4.2 Texture mapping

observing the consistency in the intensities between the two features.

The current implementation does not support the identification of the actual surfaces, so the texture is maintained by the feature pairs that correspond to the bounds of the surface. It is possible to assign the texture to each of the cells on or near the line between the two features, but will result in large amount of redundant information unless the camera does not correctly pick out the surface boundaries.

The texture that is stored is kept at a minimum, as the features cannot distinguish the difference between a surface boundary and a pattern on a flat surface without analysing the depth of the feature with respect to other features. This means the features also acts as boundaries for patterns that are present on the surface, which can be combined to construct the view of the object. The use of the surface texture is left for future development, as localisation based on correlations with the map is not carried out in this project.

## 10.5 Summary

The vision processing algorithm used to observe the scene requires many different considerations to those observed by the floor pointing cameras. The reduction in the constraints means the features that are tracked can differ significantly to those expected and lead to a less reliable tracking of the same object. Instead of attempting to perfect the tracking of a selected few features, the algorithms that are incorporated attempts track multiple features that are likely to appear in the subsequent few frames that are directly applicable to the local map.

The detection of the features involves several scores being established to identify characteristics that represent a boundary region between two surfaces of different textures. This requires multiple colour scales to minimise lighting changes, as well as comparison with neighbouring pixels to distinguish between noise and an actual surface boundary. During this process, the left and right surfaces adjacent to the boundary is distinguished, such that only one surface is required to represent the boundary.

Several filters are introduced to reduce the feature candidates that are deemed uninteresting or redundant. The strategy is combined with a range based threshold to limit the total number of features that are maintained, as well as controlling the triggering of identifying new features. This technique is used to maintain a constant pool of features that are to be tracked, but has an upper bound to guarantee the upper limit on the execution time.

After the features are tracked by correlating the intensities and determining the presence of a boundary, they are stored and used to establish the depth measure of the feature by combining multiple positions of the feature at various camera positions. The triangulation process is aided by the availability of a large number of position pairs that assist in reducing the aliasing and measurement errors.

Although there are limitations in the performance, the algorithms showed some consistency with the measures available on the local map, thus forms the basis for a higher level analysis that can be used to improve or assign extra attributes to the local map.

# Chapter 11 – High level map analysis

The small group of pixels that indicate a particular pattern has so far allowed for local features to be identified, which results in large number of redundant features that represent the same object. The large amount of information can be grouped through by a higher level representation, thus allowing a more informed enhancement to the existing map and a reduction in the redundant contributions (Smith et al., 2006).

Before the higher level concepts can be applied, the group of features must be combined using templates and meet certain conditions for the presence of such structures. The analysis involves the specification of such constraints, the formation and maintenance of the group, as well as the effect and application of the derived structures to the maps. Based on the current use of the maps, the focus is placed on the improvement in the efficiency of the visual processing component as well as the accuracy of the attributes that are assigned to the map.

## 11.1 Feature clusters

The selection of the features is typically done using their uniqueness and arrangement of the textures with respect to the surrounding pixels. Given that the view consists of structures of the scene, an important characteristic that ought to be included in the vision processing is the inter-feature relationships for those that belong to the same physical object. This characteristics can allow for extra constraints to be used when disambiguating the feature motions, increase in the reliability of the tracking, allowing the structural information to be used, and compress the duplicated tracking motions (Rosten & Drummond, 2005)

The two groups that can be formed using the visual features are in one of the object surface or the surface boundary categories. These are complimentary to each other as the object surfaces can be derived indirectly by reversing the candidates for the boundary features. While the detection of the surface can also be done through joining the surface boundaries, the pattern on the surface can sometimes hinder the distinction between a surface boundary and printed patterns on a flat surface. This distinction is often achieved through sensor fusion and segmentation algorithms, which observe the proximity and similarity in the intensity or fluctuations of the intensity at multiple scales (MacLean et al., 1994; Sharon et al., 2001). Another critical attribute to consider is the proximity of the features, as the physical objects it views should be joined together as one construct.

The capability of forming feature groups for morphing objects, such as people and rotating objects, is a challenging task that often requires significant amount of dedicated resources and predefined attributes to narrow down the search space. Instead, the focus is placed on forming a shape and density bound groups using the intensity and intensity transition characteristics. The structures that are determined will greatly simplify the merging process with the local maps, as the structures can



## 11.1 Feature clusters

be determined more accurately before they are applied to the map.

### 11.1.1 Boundary formation

Splitting the image into several distinct components can be achieved using the horizontal boundaries that have been determined by the features, which represent the surface boundaries. Using the intensities of the surfaces that are attached to the features, the similarity between the neighbouring features can be established.

When combining the features, the direction of the boundary and the density of the features play a significant role in correctly portraying the boundary of the actual object. As the current set of features represent the horizontal transitions, the boundary must extend vertically, and occasionally in a diagonal direction due to misalignment issues or irregularly shaped objects. Although the presence of direct horizontal neighbours has been reduced using the density mask, they may still exist after the features has moved around within the frame.

The detection of a connected vertical sequence is a trivial task, which simply requires a linear traversal between vertically adjacent features. However, this configuration of features is rarely observed and requires additional consideration for diagonal transitions, zigzag arrangement of features along a heavily interpolated and noisy boundary region, connecting together multiple segments that may have been separated due to obstruction or noise, and also the identification of the upper and lower bounds of the line.

It is possible to consider line detection algorithms like Hough transforms with restricted range in the slope of the line (Mattavelli et al., 1999), but this can be a very costly operation and does not easily allow the integration of other characteristics, such as the aliasing errors, consistency in the feature intensity, and proximity between the features that form the line. To cater for the aliasing errors, each vote for the line equation can be spread across several points, such that partial votes are given to those that have a similar line equation. Figure 11.1 illustrates this concept by incrementing the adjacent line equation entries as half of a vote, which can be seen as the blurring of the curves.

Another technique that can be applied is aimed at placing a constraint on the precision between the line equation parameters, which involves determining the slopes of all the potential feature pairs and only using those as the possible slope of the line. The selection criteria for the feature pairs can include restricting the horizontal window size and limiting the distance between the features, such that features that are too far apart do not attempt to combine together. If the approach is used by itself, the aliasing and precision errors in the feature can cause large number of unique line equations, thus require blurring like the above approach and increasing the possible slopes by including those that are derived from the adjacent positions as well.

Since the lines that are determined are applied throughout the whole image, the line based approaches can combine segmented lines indirectly. However, most approaches make no distinction between the various surfaces that the feature represent. The approach is also very sensitive to the density of the features and ignores the characteristics of non-features along the line being drawn, thus require

### 11.1.1 Boundary formation

normalisation to be able to identify multiple lines, as well as several parses using different attributes to group the lines.

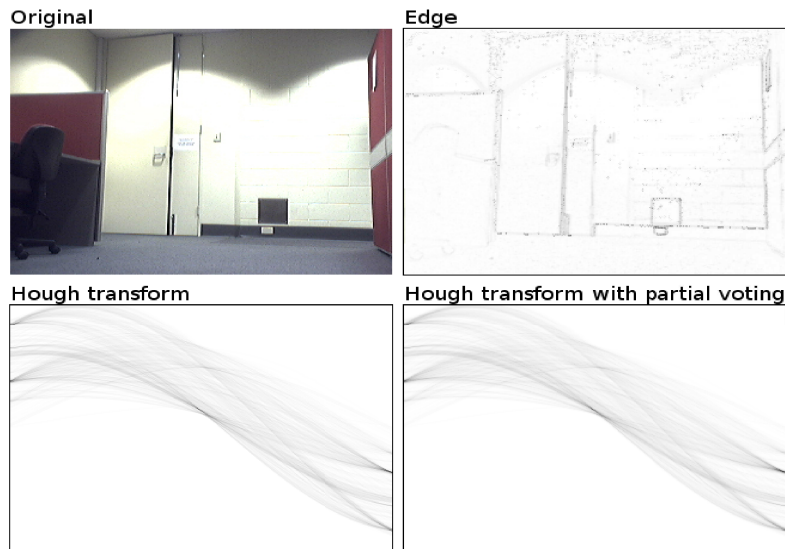


Figure 11.1: Partial voting for line detection with Hough transform. The top row shows the two sources, where the left is the original and the right is the edge image. The bottom left shows the standard Hough transform, while the bottom right shows the result of partial voting. The curves show a slight blur due to account for minor misalignments of the edges.

Based on the idea of directly joining of the features, they can be traversed recursively by placing a constraint to the direction of traversal to the same slope as previously established. If an adjacent feature is identified, but do not meet the slope criteria, a new line can be traversed from that point on.

By severely limiting the angle to those specified above, the only possible lines will be a directly vertical line, and two lines that have a slope of  $\pm 1$ . By specifying a range of angles using the corner points of a virtual pixel around the feature's position, a wider range of line slopes can be accepted. Figure 11.2 illustrates this idea by incrementally reducing the valid slope range and comparing it with the above.

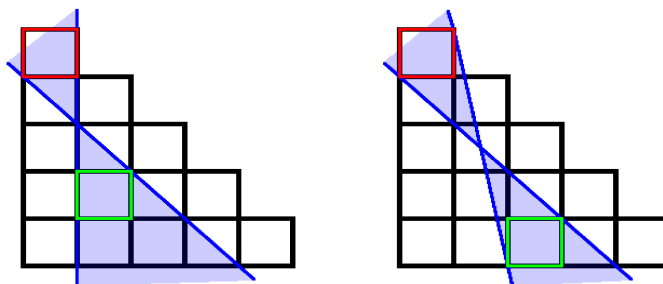


Figure 11.2: Line slope convergence through iterative line tracing. The red squares show the starting pixel and the green squares show the current pixel being observed. The blue region shows the possible slopes, which slowly decreases in area as the green square moves downwards.

### 11.1.1 Boundary formation

One of the issues with the approach, as can be seen with the second to last row, is that the lack of the intercept value being maintained means the lines that are formed do not necessarily intersect all of the past features. Although this can be problematic if the boundary is perfectly straight, the flexibility can actually be beneficial in connecting the scattered features together.

Since this approach is based on drawing a line from a specific starting feature, it is quite simple to restrict the features that can be combined. As the line is extended, the left and right intensity values that are stored can be used to determine the similarity in the surface. The approach that is currently used is a sum of the intensity difference based threshold, but by relying on just the starting feature, it may form shorter lines if the starting value was slightly corrupted due to interpolation. To account for this, the minimum and maximum intensities are stored along the line, where the difference between the minimum and maximum must remain within the threshold value. This concept is illustrated in figure 11.3 using a grey scale intensity.

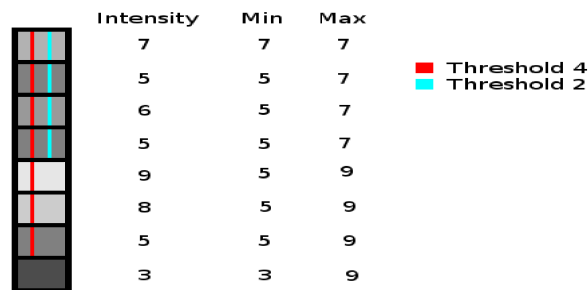


Figure 11.3: Minimum and maximum difference threshold. The traversals with different thresholds are illustrated by the red and blue lines.

Due to the reduction of features through various filters, the total number of features can be quite small when the grouping algorithm is conducted. This means the density of the features may not be high enough to form a line of any significant length. The vertical gaps between these features can either be skipped by extending the search area for the next feature to extend to, or by actually observing the intensities of the pixel that is located at that position.

The first approach can be implemented by specifying the allowed gap size and extending the triangle shaped search area until a feature is found or the limit is reached. One of the potential issues with this is if multiple features are present within the triangle, which can bias the orientation of the line based on what order the search area was traversed. By allowing the entire search area to be searched to form new lines every time a pair is found, the total number of lines can increase dramatically, as well as repeating line segments that are overlapped.

To control the number of line segments being found, the size of the triangle is trimmed down both vertically and horizontally. To remove the duplicate line segments that may be formed, any features with the same slope of a previously found feature is disregarded. Figure 11.4 illustrates the search area, where the red boxes indicate those remaining after the vertical and horizontal limits, while the grey indicates the repeated slopes. Note that these are further eliminated later on due to the constraint in the slope range, which is defined by the current state of the line. Due

### 11.1.1 Boundary formation

to the use of this mask, it is also possible to relax on the slope requirement by simply eliminating certain regions depending on the rough slope measure. This could be as simple as removing one side of the mask based on the sign of the slope.

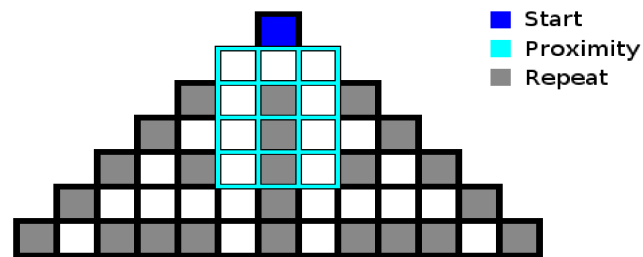


Figure 11.4: Reduction of the search area for the next feature. The top square represents the current pixel of interest. The triangle shape shows the orientation limited pixels, the grey boxes represent those that repeat the same slope as one earlier, and the outlined squares represent the pixels within close proximity.

If the line segment cannot continue, the mid point of the valid slopes can be used to define the characteristic of the line. Since there may be many short lines that are formed, it is important to eliminate these as being insignificant to the scene structure. The height of these lines also play a role in the applicability to the scene structure, as lines that are too high may not be viewable by the range finders to be included in the map. Since these lines should extend from the height viewable by the range finders, it is important that they intersect the mid-height level within the image. Rather than eliminating these lines after they have been traversed, the direction of the traversal can be specified, such that the features above the mid way point extends downwards, while those that are lower extends upwards. As these features eventually combine to indicate the start and end points of the line, it is only necessary to traverse from half of the view. A typical scene that is viewed consists of more features in the upper half of the image, as the majority of objects in the scene are located where they can be interacted easily by people and the floor textures often do not allow features to be present. This means the line formation can be started from the bottom half and extended upwards.

As the features are traversed, it is possible to maintain all the intermittent features or to eliminate them depending on the presence of multiple lines that converge at the same feature point. If the features are eliminated, the line being formed from an alternate direction which intersects the same point may be halted. If the features are maintained, then the number of line segments that are formed can potentially become an issue.

A simple approach to handle this is to observe that the lines can meet, but never intersect over another, as they represent the surfaces structure. By starting from the features that are from one end of the image, it will prioritise a longer sequence of features. When a meeting point is encountered, the line is terminated if the slope is in the opposite direction. Since the precise range of slope is not used, the direction is categorised into three rough types. If the sign of the slope is opposite to one that is already present, the feature is removed from further line analysis. Figure 11.5 illustrates the two cases using a simpler search area which only extends to the three adjacent positions starting from the top.

### 11.1.1 Boundary formation

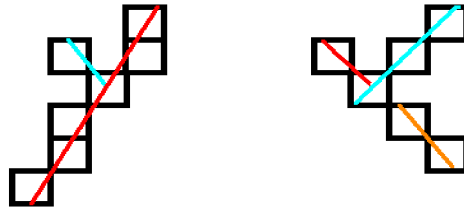


Figure 11.5: Feature group elimination for line segment formation. The lines originating from the top indicate the initial lines that are drawn. The other line is then drawn, but encounters a line with an opposite slope, thus terminates. In the second example, a third line is introduced.

The increased search area for the next feature means the pixels in between can potentially contain irregular intensities, which may be the reason for the lack of a feature to connect the line segments. It is possible to compare the intensity of the non-feature pixels with that used for the line, but as long as the search area is kept small enough, the small gaps can be noted as noise or obstructions and not from completely different surface boundaries. If the size is ever increased significantly, the consistency in the intensity and the presence of a boundary is necessary for the pixels that lie along the path between the features. It is also possible to simply increase the number of features if the resource and processing issues can be addressed by other means.

Once the lines have been defined, the features that were traversed can be combined to represent the surface boundary. The grouping allows the individual features to be influenced by the behaviour of the others within the group, which include the pose, motion, and the interactions with the local map. By introducing this reverse influencing, it can run into error propagation issues and too much narrowing of attributes, which can limit the growth and adaptation of the features. Instead, the features themselves are maintained separately and the group stores several attributes of its own, including the pose in the scene, colour, score, angular position within the image, a counter, and an id to allow referencing by the individual features that make up the group.

Since the features are constantly updated, the feature group must also be updated to account for the change in the features. When there is motion amongst the features, it is possible to make use of the feature group id to identify the motions of the other features to indicate if an unexpected motion occurs for confirmation. Depending on the reliability of the feature tracking, it may be possible to avoid this task, as it can consume valuable processing time with little benefit. If constant validation is not carried out, the groups are not used until they are applied to the map. This means the updating of the groups can be delayed until just before the features are lost or the group is used to save on the processing time.

The attributes of the group is updated when one of the feature members is removed and its state is merged with the group. The values are weighted based on the counter values of the feature and the line, which is then added together. The colour is maintained in a similar fashion, but uses a constant and equal weight for both sides of the boundary. The score relates to the number of features that make up the line, which, in conjunction with the counter, can be used as a confidence measure when

### 11.1.1 Boundary formation

the line is applied to the map.

When features are removed, the pose, counter, colour, and score must be updated. It may be possible to translate the current state of the feature group to the local map, but this is left until the members of the group are all lost in the current implementation. When this occurs, the pose and the confidence measure is applied to the local map to indicate the presence of an obstacle at that location. This is equivalent to the converged pose, which is the state at which the last feature is lost.

When new features are introduced due to the total number of features dropping below a minimum value, the feature group attributes are temporarily copied and the boundary detection algorithm is applied. As the lines are being formed, the features observe a change in the id number, which is simply a sequence number, thus requires adjustments to the feature group reference. During the rearrangement of the features, it is possible for the feature to form a part of multiple boundaries, or multiple boundaries that merge into one. When the feature group is split, the attributes can be copied over to the new line. If multiple feature groups are combined, the attributes are chosen based on the proximity to the robot, as this indicates obscuring of the background boundary by another.

Figure 11.6 illustrates a sample local map, where the boundary poses of the individual features is shown in purple, the pose determined by the feature group is shown in yellow, and the cyan circle located in the middle of the yellow cluster represents the location of the latest pose determined by the feature group. Note that to three feature groups that are still in the view have not finished converging, thus are not included in the map.

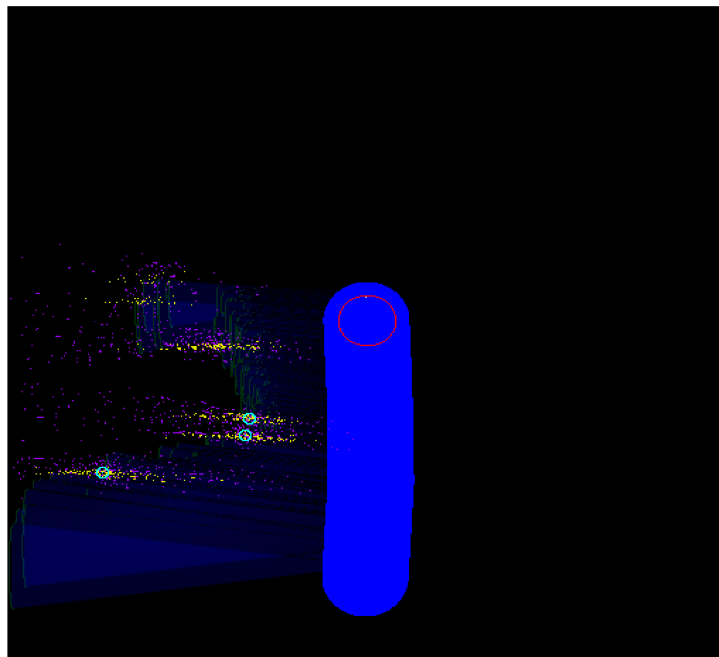


Figure 11.6: Feature group based surface boundary pose. The improved feature positions are superimposed on figure 10.15. The yellow dots represent the positions of the feature group, which converges to the cyan circle before completely exiting the view.

## 11.1.2 Surface formation

### 11.1.2 Surface formation

The use of the features allows for the corner points of the surfaces to be determined more accurately, which can indirectly be used to identify the flat surfaces that are present in between these points. However, this assumption does not hold all the time, as it relies on the precise and continuous detection of the surface boundaries. The features are currently configured to determine the boundaries of constant intensity boundaries, which do not always capture all of the vertical boundaries. This may include segments that blend in with the background, regions with rapid intensity changes that appear as noise, or not being distinctive enough and is eliminated for other, more prominent, features.

#### 11.1.2.1 Intensity similarity

By using the intensity value as the measure of consistency, it is a simple process to cluster the neighbouring pixels after specifying a range of similar values. Depending on what value is used as the reference intensity to cluster to, the features can be grouped into multiple groups as the threshold value may overlap with one another. By parsing the image using various reference values, it is possible to determine the optimal reference values for a given threshold range to minimise the overlap or the number of clusters.

One strategy in achieving a reasonable clustering using a small number of iterations is to make use of a histogram to note the trends in the intensity distribution (Novak & Shafer, 1992; Stricker & Swain, 1994). By using the maxima as the initial reference value, a region filling algorithm can be used to expand out to those with similar intensity values. When the groups are formed using an intensity based similarity, the lighting effect can cause gradual intensity differences, thus the similarity measure must be carried out using the intensity difference between the neighbours and not the bounds of the histogram. To prevent excessive grouping, the threshold value must be reduced to minimise the erroneous grouping of those with similar intensities.

To remove the occurrence of multiple associations, the pixels can simply be removed from the histogram once they have been placed into a cluster. Note that this approach may not allow large clusters to form if the thresholds are too small, as it does not consider the frequency of the neighbouring intensities. Since the intensities of the same surface should be quite similar to each other, especially with regards to the hue values, the histogram can be based on the hue scale and also make use of the adjacent frequency count when determining the most common colour. An alternate approach to this is by reducing the precision of the hue value, which allows for slightly faster processing but can cause non-optimal grouping. To improve the effectiveness of the grouping, the range of the values can be taken into consideration to stretch out the range or to modify the number of neighbours to consider for better utilisation of the available colours.

As noted earlier, the hue scale is not an effective way to identify the surface colour if the colour is close to a grey scale colour. Instead of using the hue scale, these colours are grouped together by the luminance values if the difference between the intensities is within a threshold value, which is determined from the ambient

### 11.1.2.1 Intensity similarity

noise level for the camera. Note that since the hue scale removes the gradual shading effect from lighting, the threshold should be fixed for the hue groups and adaptive for the luminance groups.

Once the first cluster has been identified and those involved removed from the histogram, the next most frequent hue or intensity group can be selected and a random pixel from the group can be selected as the starting point for the next region filling algorithm. The pseudo-code for the grouping algorithm using the sliding window is illustrated in algorithm 11.1. Note that only the luminance group is shown, as the hue group formation is done in the same way with the exception of the wrapping that occurs between the two ends of the array and the fixed threshold.

```
function GroupFeatures(image):
    set hue_image[image.height][image.width] = { -1... }
    set lum_image[image.height][image.width] = { -1... }
    ConvertToHueLuminance(image, hue_image, lum_image)
    set lum_histogram[range]
    PopulateLuminanceHistogram(lum_image, lum_histogram)
    set group[]
    GroupLuminance(group, lum_image, lum_histogram)

function ConvertToHueLuminance(image, hue, luminance):
    for h in 0 to image.height:
        for w in 0 to image.width:
            set Imax = max(image[h][w].R, image[h][w].G,
                image[h][w].B)
            set Imin = min(image[h][w].R, image[h][w].G,
                image[h][w].B)
            if Imax - Imin < ambient noise threshold:
                set luminance[h][w] = (Imax + Imin) / 2
            else:
                if Imax == image[h][w].R:
                    set hue[h][w] = (image[h][w].G -
                        image[h][w].B) / (Imax - Imin) / 6 * range
                    if hue < 0:
                        set hue[h][w] = hue[h][w] + 1
                else if Imax == image[h][w].G:
                    set hue[h][w] = (2 + (image[h][w].B -
                        image[h][w].R) / (Imax - Imin)) / 6 * range
                else:
                    set hue[h][w] = (4 + (image[h][w].R -
                        image[h][w].G) / (Imax - Imin)) / 6 * range

function PopulateLuminanceHistogram(luminance, histo):
    for h in 0 to image.height:
        for w in 0 to image.width:
            set value = luminance[h][w]
            if value == -1:
                continue
            set histo[value].pos[histo[value].count].x = w
            set histo[value].pos[histo[value].count].y = h
            set histo[value].count = histo[value].count + 1
    histo[0].score = histo[0].count + histo[1].count
    for v in 1 to range - 1:
        histo[v].score = histo[v - 1].score +
            histo[v + 1].count - histo[v - 1].count
    histo[range - 1].score = histo[range - 2].score -
        histo[range - 2]
```



### 11.1.2.1 Intensity similarity

```
for v in 0 to range:
  if histo[v].count == 0:
    histo[v].score = 0

function GroupLuminance(grp, luminance, histo):
  while true:
    set maximum = 0
    set index = 0
    for v in 0 to range:
      if histo[v].score > maximum:
        maximum = histo[v].score
        index = v
    if maximum == 0:
      break
    else:
      set x =
        histo[index].pos[histo[index].count - 1].x
      set y =
        histo[index].pos[histo[index].count - 1].y
      set stack[] = { { x, y } }
      while stack.size > 0:
        set stack.size = stack.size - 1
        set w = stack[stack.size].x
        set h = stack[stack.size].y
        set val = luminance[h][w]
        set luminance[h][w] = -1
        set grp[grp.size].pos[grp[grp.size].count].x = w
        set grp[grp.size].pos[grp[grp.size].count].y = h
        set grp[grp.size].count = grp[grp.size].count + 1
        set small = 0
        set large = histo[val].count - 1
        while small <= large:
          set mid = small + (large - small) / 2
          if histo[val].pos[mid].y > h:
            large = mid - 1
          else if histo[val].pos[mid].y < h:
            small = mid + 1
          else if histo[val].pos[mid].x > w:
            large = mid - 1
          else if histo[val].pos[mid].x < w:
            small = mid + 1
          else
            small = mid
            large = small - 1
        histo[val].count = histo[val].count - 1
        if histo[val].count == 0:
          histo[val].score = 0
        else:
          histo[val].score = histo[val].score - 1
        if value > 0:
          histo[val - 1].score = histo[val - 1].score - 1
        if val < range - 1:
          histo[val + 1].score = histo[val + 1].score - 1
        for j in -1 to 1:
          for i in -1 to 1:
            set neighbour = luminance[h + j][w + i]
            if neighbour != -1 && | val - neighbour | <
              luminance_threshold:
              set stack[stack.size].x = w + i
```

### 11.1.2.1 Intensity similarity

```
set stack[stack.size].y = h + j
set stack.size = stack.size + 1
set grp.size = grp.size + 1
```

Algorithm 11.1: Surface clustering based on similarity in the intensity.

Although the clusters are given a range of intensities that are accepted as being the same, there are frequent occurrences of noise and obstruction that can split two very closely located clusters. This issue can be handled in a similar way to the boundary formation process by increasing the search area for the similar colour. Instead of simply allowing any disjointed pixel to be merged together, two extra constraints are placed to prevent different surfaces from merging together.

The first strategy involves making the intensity similarity stricter by reducing the range of accepted value as the search area is increased. This allows the two regions being combined to contain small amount of variety, such as a thin line, and still be classified as the same surface. The second strategy involves the limiting of the direction in which the search area is increased, which is based on the idea that many indoor structures are decorated with the same theme and placed one after the other. This means extending the search area sideways can potentially include surfaces that belong to other objects. With this in mind, the search area is only extended vertically, where the allowed colour range is halved for every pixel being extended until the maximum possible extension, which is currently set to 5 pixels based on a rough assumption that the gap of 5 mm is allowed when observed at a distance of around 1m.

### 11.1.2.2 Texture pattern

When the surfaces are viewed at different distances, various texture patterns may appear or disappear due to different sampling rates by the camera. When these texture patterns are visible, it can sometimes trigger a boundary feature to be established, as the inconsistent intensity prohibits the intensity similarity based surface detection from operating. To successfully group the pixels that are from the same surface, techniques such as multiple resolutions, template based, or intensity fluctuation based techniques can be applied to characterise the arrangement of the pixels.

The use of multiple resolutions is done to simulate the change in the distance to the scene by interpolating the neighbouring pixels, which allows the suppression of rapidly changing intensities. Since the change in the actual camera resolution can be a slow process due to the re-initialisation of the camera device, this can be simulated in software by a sub-sampling algorithm. To find the appropriate interpolation strength and size to use, the frame must be processed multiple times with various values until the number of segments that are formed reaches a desired level. The settings for the sub-sampling algorithm does not always result in the ideal surfaces being determined, as the spacing between the patterns can vary from texture to texture. It is possible to use different resolutions on different portions of the image, as with many segmentation algorithms, but these tend to consume a significant amount of processing time due to the large amount of search space, so are not carried out here.

### 11.1.2.2 Texture pattern

Instead of attempting to convert the rough textured pixels into a uniform intensity region, the intensity pattern itself can be observed. Two of the characteristics that are common to the majority of non-uniform intensity surfaces are the presence of repeated texture patterns and a very similar rate of fluctuation in the intensity. The detection of these requires a group of pixels to be observed simultaneously, such that the correlations can be carried out over a region.

The process of identifying repeated patterns involves a similar idea to deriving the correlation scores between texture patterns at neighbouring positions. One of the additional considerations to make is the discarding of irregular intensities between the repeated textures, which means the size of the pattern must be modified when different sized patterns are encountered. As with the feature tracking algorithm used for localisation, the evaluation of the correlation score can be quite time consuming due to the various overlap arrangement of the pixel intensities.

To simplify some of the processing, it is possible to make use of templates to characterise the particular intensity behaviour, such as stripes or meshed patterns, such that constraints are placed on the expected arrangement as well as the distance at which it must be observed at. The templates that are used must define the the intensities or the relative intensity changes in case a shape based pattern is required.

The shapes and sizes of these templates can depend on the anticipated structures in the scene and the resource availability, which can control how precise and adaptable the patterns can be. Since the orientation and the density of the texture pattern can easily be misaligned, the templates can be applied in sequence with different configurations. Figure 11.7 illustrates a sample template that has been designed for detecting vertical stripes. Note that the difference in the intensity is measured by the magnitude to allow the inverse of the pattern to also be detected.

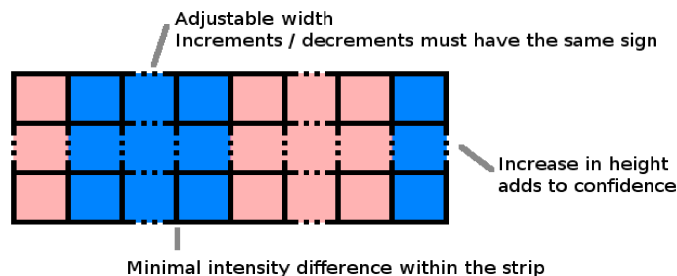


Figure 11.7: Vertical stripe detection templates.

The two colours represent the two alternating colours. The dotted lines represent multiple occurrence of the same colour.

When using these templates, it is ideal to know, or learn, that the anticipated texture patterns actually do exist in the scene. Without this information, the templates can potentially be a waste of valuable processing time. It is possible to maintain a counter to note how frequently the pattern is matched to indicate whether the template is actively being used, but this does not deal with the fundamental issue that these templates must be actively used before they can be determined that they do not suit the particular environment. The counters must also consider partial matches unless a threshold can be set for when the template is matched or not.

One strategy that can be used to counter this is the relaxation on the specific

### 11.1.2.2 Texture pattern

arrangement of the intensity transitions within the template. By expecting a certain amount of fluctuation in the intensity within the specified region, it is able to detect patterns based on the rate of intensity change over the region, rather than specifying the exact arrangement of the intensity differences. This approach is useful on randomly arranged or rough textures, such as carpets and rough surfaces under directional light. When using this approach, it is important to avoid certain locations that are known to contain high amounts of change in the intensity, but is not a continuous surface. Currently, the only indicator for this is the location of the features and to some extent, the feature candidates.

Although the above approach may seem to be reasonable, the sloped view of the ground texture and the sampling of the surface at a reasonable distance away means the majority of the fluctuation in the intensity can not be seen or appears like noise. The repeated textures on other surfaces also has to either be quite large or viewed at a very close distance, which severely limited the applicability of the real time pattern based surface recognition, thus is simply not used in the current implementation. The surface construction is also not implemented in the current maps, thus may require an additional layer to indicate the positions and texture separately if future implementations require surfaces.

The failure to note this type of surface can lead to small amounts of boundary features being detected in their place, but the vertical patterns do not cause any significant issues when the pose is applied to the local map. The motion behaviour of these features are quite distinct in that it often jumps rapidly in an inconsistent manner. These features can be removed either during the line formation phase or by simply noting the rate of fluctuations in the observed positions, which often switches the motion direction very rapidly or falls outside the search area and results in a very low correlation score.

### 11.1.3 Dynamic objects

So far, the vision based analyses has presumed that the scene remain stationary between consecutive frames. However, there are often many independent objects within the environment that can move about to cause motions that do not correspond to the pose change of the camera. Since the structure of the scene is analysed through the different perspectives from known poses, the detection of dynamic behaviour must be conducted by either knowing the pose of the feature or an approximation of the object's pose can be made based on constraints provided by the camera poses.

The first approach can be a useful technique if the pose of the object can be determined through other means, such as a range finder or a stereo vision configuration. Otherwise, the precise pose of the dynamic objects cannot be tracked simultaneously and reverts back to the second approach. By using the constraints provided by the camera pose and sometimes based on simple assumptions, such as the expected size, shape, and velocity of the dynamic object, they can be isolated from the static structures in the view.

When the dynamic objects are observed from a per pixel perspective, the entire object may not appear to have moved depending on the texture on the surface. Since the many of the pixels surrounding the boundary will be flagged as having changed,

### 11.1.3 Dynamic objects

segmentation or contour following algorithms can be used to define where the dynamic object lies in the image.

To identify the area of interest, it is firstly important to distinguish the change in the appearance between the camera motion and object motion. As noted earlier, the pose of the robot may be slightly out of sync depending on the light conditions, which means the distinction between the two motions should be carried out using the same camera when possible or delayed until the pose data becomes available. One implementation of this is to observe that portions of the current features are moving irregularly compared to the rest. Since the only association that is available between the features is the boundary groups, the distinction between the dynamic object, the background, and any noise, requires a separate process to identify the presence and the grouping of dynamic objects.

A simple case for this is when the camera is known to be stationary, thus any large cluster of features that move by more than 1 pixel, due to interpolation, can be flagged as being dynamic. When this is carried out, it is important to observe the possible latency issue between the camera pose update and the changing of the view. This can be achieved by delaying the dynamic object detection by several frames to make sure that the pose of the camera does not change. This constraint also allows simple analysis to determine the direction of the motion and the bounds of the dynamic objects. However, since the pose of the dynamic objects cannot be determined precisely without referring to the local map, the region can only be flagged as containing non-static objects.

As noted earlier, these features can sometimes be lost due to the reduced search area, thus is unable to account for a wide range of possible objects that may appear in the view. The viewing area also restricts how useful the tracking is unless the facility to track a particular object is incorporated to the servo motors. The handling of the dynamic objects that are detected varies significantly on the application, ranging from pose tracking to observing the morphological and inter-dynamic object changes. The flagging of the dynamic objects can assist in the current implementation by noting the possible areas that cannot be used later on for correlation, as well as marking areas of potential obstruction and vacancy depending on the current pose of the dynamic object. The application and implementation of this will be discussed in more detail later on when using the omnidirectional vision camera.

## 11.2 Landmarks

The added benefit of using a wide viewing area provided by the webcam is its ability to identify a significant pattern that can be uniquely identified at a later time. This can be used as calibration markers to correct any drifting errors, as well as providing a meaningful reference location to be tagged at a higher level map (Sim, 1998). For the ground observing camera, it is important to involve a large amount of pixels to determine the uniqueness of the landmark, while the side viewing camera only needs to analyse a small amount of pixels to identify a unique arrangement of pixel intensities. This is due to the repetitive and detailed nature of the ground textures, which requires not only an increased size in the area to consider, but the presence of a significantly different ground texture to be present.

### 11.2.1 Ground landmarks

#### 11.2.1 Ground landmarks

By reusing the images from the localisation module, a larger viewing area can be observed to determine the uniqueness of the ground texture with respect to a larger range of possible ground textures. Because of the similarity between the floor textures, the attributes that are used to distinguish the particular texture pattern must be taken when a different texture pattern comes into view. This is due to the weaker set of constraints that can be applied, especially due to transformations from rotation.

The process of identifying the landmark involves a similar process to the regular feature tracking, but includes a decision in determining if a landmark should be captured or not. As one of the criteria for the landmark formation is in observing a large change in the trend of the ground texture, the segmentation algorithms introduced above can be applied to identify the presence of distinctive regions (Zhang & Kodagoda, 2005).

Due to the clarity of the ground textures from the small and fixed distance to the surface, it is possible to make use of the rate in the change of intensity, but this requires an increase in the size of the convolution kernels, which can require multiple passes until the appropriate pattern is found. Instead, the rapid rate of changes can be suppressed using a blurring algorithm such that the analysis can focus on the difference in the intensity using a simple and small kernel (Lindenberg, 1996). The loss of the precise position of the boundary can be addressed by reverting back to the original image once the initial filtering of boundary identification is complete, while the prevention of larger patterns appearing as unique boundaries can be solved by noting the direction of traversal of the robot, such that only the first of the change in the intensity encountered will be flagged as the landmark candidate. However, using this approach can also prevent actual boundaries from being detected if false candidates or corrupted texture pattern is viewed, such as when motion blur occurs. For this reason, the viewing area for the landmark candidates is fixed to increase the chance of detecting the landmark.

The second stage of the landmark detection involves the assumption that the new segment being introduced is large enough that it spans across the viewing area. This means the change in the appearance will be visible at at least two of the outer borders of the image. By isolating the initial check to the boundaries, it can quickly determine whether any interesting texture has entered into view. Since the segment should occupy a reasonable amount of area, the corner regions of the image are avoided to suppress small segments that only appear in the corner and to wait for the it to move into a more central position within the frame. The areas used for the initial check is shown in figure 11.8 as the red regions. Note that the region is moved inwards from the boundary by a small amount to avoid the distorted areas of the captured image.

The criteria for flagging the presence of a possible landmark involve directional convolution kernels in the parallel direction to the edge it runs along to identify the dividing line that spans across the image. Before the third stage of the analysis can be carried out, the pixels containing a small amount of intensity transition must be filtered out, while the strong intensity transition must be maintained, along with the surrounding pixels to account for aliasing and misalignment errors. If an absolute value is used as the threshold, the algorithm may detect too many or too little

### 11.2.1 Ground landmarks

landmark candidates depending on the type of surface the robot is traversing over. Since the measure of uniqueness is based on the relative difference between the surroundings, a running average of the intensity difference is maintained to observe a sudden spike or trough in the value. This is achieved by maintaining two sliding windows to evaluate the average along side the point of interest, as shown in figure 11.9, where the blue and magenta regions represent the adjacent textures on the left and right respectively, the cyan, lavender, and light blue representing the maximum, minimum, and the average of the left texture region, while yellow, pink and orange represent the same values, but for the right region. The conditions shown to the side represent the detection criteria for the change in the surface texture.

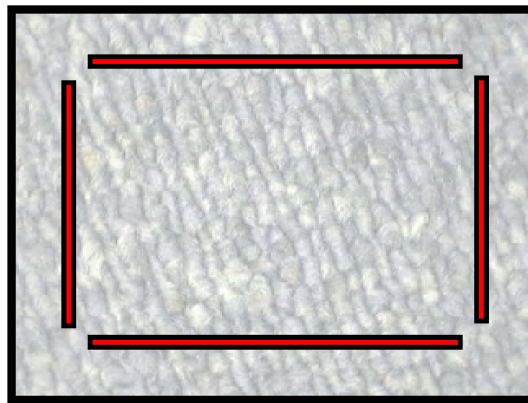


Figure 11.8: Areas that are initially considered for segment identification. The red strips represent the regions that are considered for the detection of segment boundaries.

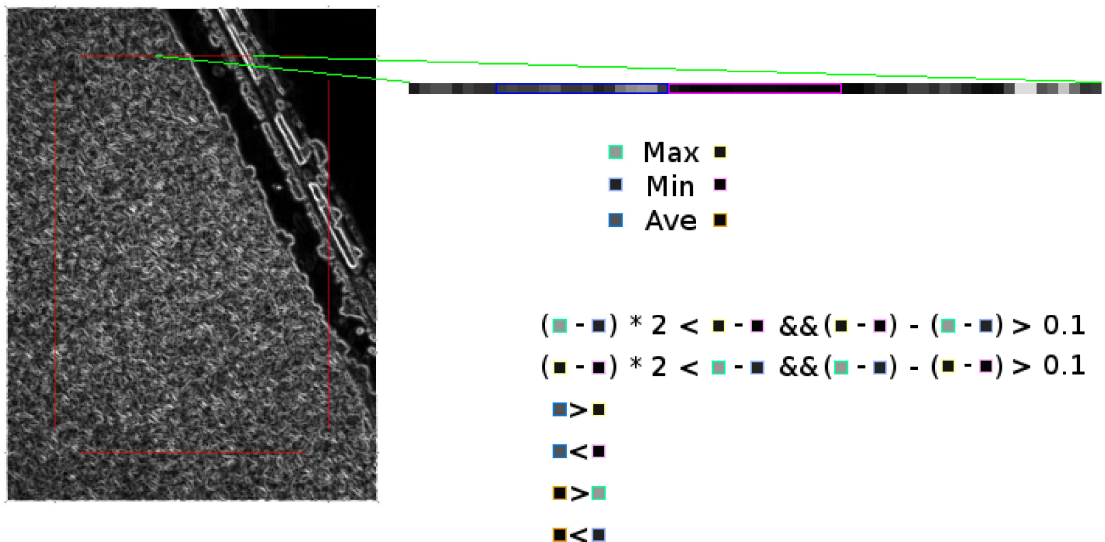


Figure 11.9: Conditions to note change in texture patterns. The red lines on the left image represent the areas being considered. The blue and pink rectangles on the top right shows the left and right regions being observed, which results in the Max, Min and Ave values shown in the middle-right. The comparisons on the bottom right are used to determine if there is a change in the surface boundary.

### 11.2.1 Ground landmarks

Instead of complicating the process with values like the standard deviation and second derivatives, the minimum and maximum within the window is maintained to note when the distance between the two changes suddenly. In the current implementation, a threshold value of 50% change has been specified, along with a minimum threshold of 10% of the maximum range. This allows some elimination of false flagging if the change is too small, which may have been caused by random noise. As for the average values, the flagging is carried out if the average value falls outside the maximum and minimum range. The width of the window has been set to 16 pixels, which equates to approximately 4.6 mm when using a resolution of 320 x 240. This may be too small or large to observe some of the texture patterns, but is a reasonable amount when observing the typical textures on the ground.

The assumption that the landmark extends across the whole image is used to note the continuity in the change in the texture by joining two sides that have both been flagged as containing the boundary. The points can be used to trace a line between them to observe whether the change in the ground texture is consistently present. Instead of tracing narrow lines between the points, the thickness of the line is increased to include the adjacent pixels. This caters for the aliasing and misalignment issues, as well as allowing for a small amount of flexibility in the straightness of the boundary, which may be caused by faded lines or chipped edges.

The line that is traced places a strict constraint on the shape of the landmark which is required to have a straight edge to fall within the viewing area. This can be rectified by using a similar tracing algorithm used for the vertical line segments, where a triangular search area is used to continually trace through the neighbouring pixel with a change in the intensity. To account for the start and end points, the search can be executed simultaneously from both ends, like a means-end search algorithm. However, since the viewing area is quite small and the majority of the significant dividers between floor segments are shaped using straight lines, the flexibility in the curve does not provide much of an additional benefit and a simple line tracing algorithm is used.

During the traversal of the lines, the edge strength in all surrounding directions are summed to indicate whether a strong change in the intensity occurs between the two points. The average of the edge strength can be used to determine if a line can be drawn which separates the two regions. The equation of the line can be based on the two end points, which may not be the precise location of the boundary, but still allows coverage of the boundary due to the expansion of the line thickness.

Although the lines are only drawn if the end point pairs lie on different sides, it is quite possible that multiple lines can be drawn, especially along side one another due to the interpolation that occurs at the segment boundaries. By using multiple neighbouring lines, it is possible to narrow down the actual location by using the scores for the lines, but this is avoided due to the misalignment between the line equation and the actual line. Instead, the line with the highest average transition score is kept as the only segment divider. As for determining whether the line constitutes a valid landmark or not, the line should be compared against a minimum requirement to avoid incorrect marking of landmarks. Since a detailed analysis on the types of textures that the robot can encounter requires a reasonable amount of motion before it is known to the robot, this measure can be based on assumptions or the landmarks can be ranked to only maintain and use the top candidates.



### 11.2.1 Ground landmarks

Using the line equation as the landmark allows for a much simpler and reusable landmark, since using the exact texture pattern would require significant amount of processing to account for the misalignment and transformation. By compressing the landmark to a line and maintaining the information on the surrounding pixel intensities, it is able to constrain some of the pose attributes of the robot when it is encountered again. This also allows some flexibility in where the feature can be observed by assuming continuity of the line. Since the bounds of the line are unknown, it is left for the post processes to extend and possibly join these landmarks. The general flow of processes for the landmark detection can be seen in algorithm 11.2.

```
function FindLandmark(image):
    set candidates[]
    set window = 16
    set short = image.width / 10
    set long = image.height / 10 - window
    GetCandidates(image, candidates, window, long, short)
    ApplyBlur(image, candidates)
    IntensityDifference(candidates)
    ThresholdCandidate(candidates, window)
    for i in 0 to candidates[0].value.size:
        candidates[0].pos.x = candidates[0].value[i] + long +
            window - 1
        candidates[0].pos.y = short
    for i in 0 to candidates[1].value.size:
        candidates[1].pos.x = candidates[1].value[i] + long +
            window - 1
        candidates[1].pos.y = image.height - short
    for i in 0 to candidates[2].value.size:
        candidates[2].pos.x = short
        candidates[2].pos.y = candidates[2].value[i] + long +
            window - 1
    for i in 0 to candidates[3].value.size:
        candidates[3].pos.x = image.width - short
        candidates[3].pos.y = candidates[3].value[i] + long +
            window - 1
    set lines[]
    TraceLine(image, candidates, lines)
    if lines.size == 0:
        return null
    set best = 0
    for i in 1 to lines.size:
        if lines[best].score < lines[i].score:
            set best = i
    if lines[best].score > minimum_score:
        return lines[best]
    else
        return null

function GetCandidates(image, candidates, long, short):
    for y in short - 2 to short + 3:
        for x in long - 3 to image.width - long + 3:
            candidates[0].value[y - short + 2][x - long + 3] =
                image[y][x]
            candidates[1].value[y - short + 2][x - long + 3] =
                image[y + image.height - 2 * short][x]
    for y in long - 3 to image.height - long + 3:
        for x in short - 2 to short + 3:
            candidates[2].value[y - long + 3][x - short + 2] =
```

## 11.2.1 Ground landmarks

```
        image[y][x]
        candidates[3].value[y - long + 3][x - short + 2] =
            image[y + image.width - 2 * long][x]

function ApplyBlur(image, candidates):
    set kernel = { { 1, 2, 3, 2, 1 }, { 2, 5, 7, 5, 2 }, { 3, 7,
        9, 7, 3 }, { 2, 5, 7, 5, 2 }, { 1, 2, 3, 2, 1 } }
    set total = 89
    set value0[candidates[0].value[0].size]
    set value1[candidates[1].value[0].size]
    for x in 0 to value0.size:
        set sum0 = 0
        set sum1 = 0
        for j in 0 to kernel.size:
            for i in 0 to kernel[0].size:
                set sum0 = sum0 + kernel[j][i] *
                    candidates[0].value[j][x + i]
                set sum1 = sum1 + kernel[j][i] *
                    candidates[1].value[j][x + i]
        value0[x] = sum0 / total
        value1[x] = sum1 / total
    candidates[0].value = value0
    candidates[1].value = value1
    set value2[candidates[2].value.size]
    set value3[candidates[3].value.size]
    for y in 0 to value2.size:
        set sum2 = 0
        set sum3 = 0
        for j in 0 to kernel.size:
            for i in 0 to kernel[0].size:
                set sum2 = sum2 + kernel[j][i] *
                    candidates[2].value[y + j][i]
                set sum3 = sum3 + kernel[j][i] *
                    candidates[3].value[y + j][i]
        value2[y] = sum2 / total
        value3[y] = sum3 / total
    candidates[2].value = value2
    candidates[3].value = value3

function IntensityDifference(candidates):
    set diff0[candidates[0].value.size - 2]
    set diff1[candidates[1].value.size - 2]
    for i in 0 to candidates[0].value.size - 2:
        set diff0[i] = | candidates[0].value[i + 1] -
            candidates[0].value[i] | + | candidates[0].value[i + 1]
            - candidates[0].value[i + 2] |
        set diff1[i] = | candidates[1].value[i + 1] -
            candidates[1].value[i] | + | candidates[1].value[i + 1]
            - candidates[1].value[i + 2] |
    candidates[0].value = diff0
    candidates[1].value = diff1
    set diff2[candidates[2].value.size - 2]
    set diff3[candidates[3].value.size - 2]
    for i in 0 to candidates[2].value.size - 2:
        set diff2[i] = | candidates[2].value[i + 1] -
            candidates[2].value[i] | + | candidates[2].value[i + 1]
            - candidates[2].value[i + 2] |
        set diff3[i] = | candidates[3].value[i + 1] -
            candidates[3].value[i] | + | candidates[3].value[i + 1]
```

## 11.2.1 Ground landmarks

```
- candidates[3].value[i + 2] |
candidates[2].value = diff2
candidates[3].value = diff3
```

```
function ThresholdCandidate(candidates, window):
  set filter[]
  set left
  set right
  for loop in 0 to candidates.size:
    for i in 0 to window:
      set value = candidates[loop].value[i]
      if left.max < value:
        set left.max = value
      else if left.min > value:
        set left.min = min
      set left.sum = left.sum + value
      set value = candidate[loop].value[i + window]
      if right.max < value:
        set right.max = value
      else if right.min > value:
        set right.min = min
      set right.sum = right.sum + value
    set left.diff = left.max - left.min
    set ave_left = left.sum / window
    set right.diff = right.max - right.min
    set ave_right = right.sum / window
    if 2 * left.diff < right.diff &&
      right.diff - left.diff > 0.1 ||
      2 * right.diff < left.diff &&
      left.diff - right.diff > 0.1 || ave_left > right.max ||
      ave_left < right.min || ave_right > left.max ||
      ave_right < left.min:
      set filter[loop].pos[filter[loop].pos.size] = 0
      set filter[loop].pos.size = filter[loop].pos.size + 1
      set filter[loop].pos[filter[loop].pos.size] = 1
      set filter[loop].pos.size = filter[loop].pos.size + 1
    for i in window to candidates[loop].value.size - window:
      set rem_left = candidates[loop].value[i - window]
      set value = candidates[loop].value[i]
      set add_right = candidates[loop].value[i + window]
      set start = i - window + 1
      if rem_left == left.max:
        for j in 0 to window:
          if left.max < candidates[loop].value[start + j]:
            set left.max =
              candidates[loop].value[start + j]
      else if rem_left == left.min:
        for j in 0 to window:
          if left.min > candidates[loop].value[start + j]:
            set left.min =
              candidates[loop].value[start + j]
      if value > left.max:
        left.max = value
      if value < left.min:
        left.min = value
      set left.sum = left.sum - rem_left + value
      if value == right.max:
        for j in 0 to window:
          if right.max < candidates[loop].value[i + j]:
```

### 11.2.1 Ground landmarks

```
        set right.max = candidates[loop].value[i + j]
else if value == right.min:
    for j in 0 to window:
        if right.min > candidates[loop].value[i + j]:
            set right.min = candidates[loop].value[i + j]
if add_right > right.max:
    right.max = add_right
if add_right < right.min:
    right.min = add_right
set right.sum = right.sum - value + add_right
set left.diff = left.max - left.min
set ave_left = left.sum / window
set right.diff = right.max - right.min
set ave_right = right.sum / window
if 2 * left.diff < right.diff && right.diff -
left.diff > 0.1 || 2 * right.diff < left.diff &&
left.diff - right.diff > 0.1 || ave_left > right.max
|| ave_left < right.min || ave_right > left.max ||
ave_right < left.min:
    if filter[loop].pos[filter[loop].pos.size - 1] !=
start:
        set filter[loop].pos[filter[loop].pos.size] =
start
        set filter[loop].pos.size =
filter[loop].pos.size + 1
    set filter[loop].pos[filter[loop].pos.size] =
start + 1
    set filter[loop].pos.size =
filter[loop].pos.size + 1
for loop in 0 to candidates.size:
    candidates[loop].value = filter[loop].pos

function TraceLine(image, candidates, lines):
    set cache[2]
    for loop1 in 0 to candidates.size - 1:
        set x1 = candidates[loop1].pos.x
        set y1 = candidates[loop1].pos.y
        for loop2 in loop1 + 1 to candidates.size:
            set x2 = candidates[loop2].pos.x
            set y2 = candidates[loop2].pos.y
            set deltax = x2 - x1
            set deltay = y2 - y1
            set score = 0
            if | deltax | > | deltay |:
                if deltax > 0:
                    set short = 1
                else:
                    set short = -1
            set deltax = | deltax |
            set deltay = | deltay |
            set error = deltay
            set x = x1
            for y in y1 to y2 + 1:
                set score = score + GetEdgeScore(image, y, x,
cache) + GetEdgeScore(image, y, x - 1, cache) +
GetEdgeScore(image, y, x + 1, cache)
                set error = error - deltax
                if error < 0:
                    error = error + deltay
```

### 11.2.1 Ground landmarks

```
        set x = x + short
    set count = deltax
else:
    if deltax > 0:
        set short = 1
    else:
        set short = -1
    set deltax = | deltax |
    set deltax = | deltax |
    set error = deltax
    set y = y1
    for x in x1 to x2 + 1:
        set score = score + GetEdgeScore(image, y, x,
            cache) + GetEdgeScore(image, y - 1, x, cache) +
            GetEdgeScore(image, y + 1, x, cache)
        set error = error - deltax
        if error < 0:
            error = error + deltax
            set y = y + short
        set count = deltax
    set lines[lines.size].score = score / count
    set lines[lines.size].pos = { x1, y1, x2, y2 }
    set lines.size = lines.size + 1

function GetEdgeScore(image, y, x, cache):
    if cache[0][y][x] == miss:
        set cache[0][y][x] = | image[y][x] - image[y][x - 1] |
    if cache[0][y][x + 1] == miss:
        set cache[0][y][x + 1] = | image[y][x + 1] -
            image[y][x] |
    if cache[1][y][x] == miss:
        set cache[1][y][x] = | image[y][x] - image[y - 1][x] |
    if cache[1][y + 1][x] == miss:
        set cache[1][y + 1][x] =
            | image[y + 1][x] - image[y][x] |
    return cache[0][y][x] + cache[0][y][x + 1] +
        cache[1][y][x] + cache[1][y + 1][x]
```

Algorithm 11.2: Pseudo-code for landmark detection.

Frequent activation of the above check should not be carried out to avoid the expensive processing load and also the overloading of the number of landmarks. To avoid the same landmark from being detected multiple times during the same transition, a cooling down period is introduced once a landmark has been spotted. The cool down period involves the waiting until the landmark moves outside of the view before another landmark is looked for again. Since the equation of the line is known, the camera pose can be used to determine the minimum distance the robot must traverse to avoid observing the same landmark.

Based on the general direction of traversal for one camera, which is along a single axis, the displacement can be accumulated to determine if the landmark has moved out of view without having to continuously track its motion. It is possible that this technique can result in miscalculating the line's motion and not waiting long enough, especially for the case where the line extends in the same direction as the traversal. However, the inclusion of the same line does not carry any negative effect other than consuming some time to identify the landmark and requiring a separate allocation of

### 11.2.1 Ground landmarks

memory if they are not combined. This means that as long as the cool down period is made large enough to ease the processing load, the landmarks can be taken frequently to encourage the merging and to provide more points in the map where the pose can be realigned. At the same time, it is important to maintain some distance between the landmarks, in case they require disambiguation.

Another source of delay is introduced to the frequency of landmark check. Since a landmark being captured in a more central position of the viewing area is encouraged, as it increases the number of pixels being traced to determine the presence of the line as well as avoiding the warping from the image distortions. This means if no landmark candidate is found, the ground texture may move up to half of the distance between the searched areas before being observed again.

Using the maximum velocity of the robot, the frame rate, and the viewing area, the maximum displacement of the landmark can be determined for it to be outside the viewing area to the middle of the image, which evaluates to approximately 3.9 frames. This means the landmark check only needs to be carried out once in four frames, as slightly overshooting the middle of the image will still allow the landmark to be visible. Although the robot may almost constantly traverse at its maximum velocity, using a fixed time interval can result in a lot of unnecessary checks for the landmark. Since the cool down period is calculated to avoid the repeated observation of the landmark, the same condition can be used when no landmark is found.

As a side note, an alternate approach of using the four corner points of the view to initiate an intensity based segmentation using filling algorithm was briefly considered. This involved the comparison of the meeting points between the boundaries, but was quickly discarded as processing requirement was quite high and it relied heavily on a strong blurring algorithm to even out intensity fluctuations between patterns on the ground.

### 11.2.2 Other landmarks

A more commonly seen example of landmarks being used is one that identifies distinct appearances at any elevation, such that the arrangement of these landmarks can inform the robot of its current pose (Montemerlo, 2002; Se et al., 2002). Rather than focusing on finding recognisable features, a specific type of feature is searched for to avoid traversing into hazardous locations. In this particular application, collision with solid obstacles and holes the robot could fall down are both hazards that must be avoided for the safe and continuous operation of the robot (Jenkin & Jepson, 1994).

The majority of the collision aspect can be avoided through the range finders, with the exception of the modules that are mounted high, which can collide with obstacles like an archway. However, there is no sensor pointing towards the ground at the front of the robot to indicate where it is about to move on to. Since the default orientation of the scene viewing camera is to the side of the robot, it is unable to effectively inform the robot of the incoming danger. One possible way to identify the presence of potential surfaces discontinuity is to mark these locations on the map if they are encountered.

The detection of these edges begins with observing horizontal, or near horizontal

### 11.2.2 Other landmarks

lines within the lower portion of the image. This can either be based on intensity changes or through the use of a segmentation algorithm. Since the texture pattern on the ground can contain patches with different intensities, a strong blurring algorithm can be applied to even out the intensities. These lines can be detected using similar approaches to the vertical line detection with slightly different sets of constraints. With the line identified, it can be categorised into one of three based on the altitude of the obstacle adjacent to the line, which are positive, negative, and neutral elevation from the normal surface.

If the lines are from objects on top the surface, the objects will appear on the map constructed by the ranger finders, thus can be discarded as being a harmless pattern on the object. The approximate location of the line in the map can be found using the following distance calculation.

$$\text{Distance to line} = \text{Camera height} * \text{Vertical resolution} / ((\text{Vertical resolution} - 2 * \text{Vertical position}) * \tan(\text{Viewing angle})) \quad (56)$$

If the line is on the same elevation as the surface, such as those from markings or boundary between different surfaces, these too can be discarded, but the difficulty lies in disambiguating the difference between the elevations of the surface beyond the line. As briefly noted earlier, determining the altitude of features through motion requires features to be tracked while the camera changes its pose. Due to the low level of precision and the reduced exposure of the feature moving towards or away from the camera, the altitude of features is more difficult to determine.

Since these regions of potentially hazardous is difficult to analyse using the forward looking camera, the distinction between hazardous and non-hazardous region can be delayed until another sensor is observing the region. Out of the current array of sensors, the only sensors that can reliably observe the characteristics on the ground are the downward looking cameras used for localisation. Using these sensors, the displacement of the ground before and beyond the boundary can be measured separately to note any large discrepancy between the two. The robot should remain on the known surface, thus the motion of the ground closer to the robot will be as expected, but the motion beyond the bound should either be smaller or equal to the other.

Note that due to the fixed focus control of the cameras, the texture of the surface beyond the boundary may be heavily blurred and may not contain many features to be tracked reliably. To counter this, the area of the surface being tracked beyond the boundary may need to be increased to allow a more reliable tracking of the ground texture. Figure 11.10 illustrates the motion tracking. The figure on the left is taken from the left camera, while the right is taken from the right camera where it is viewing over an edge.

To be able to carry out this process, the robot must navigate to just before the boundary and rotate around to observe both sides of the boundary. Since this motion behaviour will force the navigation behaviour of the robot to change, the position of the line can be stored in memory to be referenced if the robot moves over the region in the future. Another way to consider this is to ignore the horizontal line detection all together and simply carry out the depth detection around all ground landmarks that are encountered.

### 11.2.2 Other landmarks

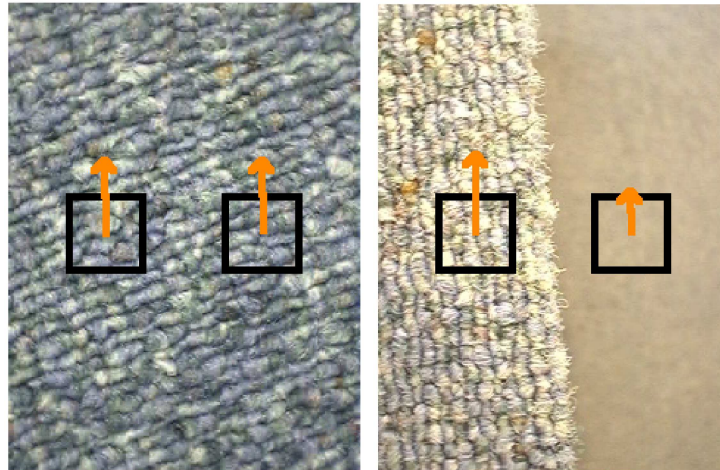


Figure 11.10: Regions for displacement comparison to note change in altitude. The black squares represent the features, while the red arrows indicate the motion vector of the features.

Since the localisation module makes use of multiple trackers for error handling, this can be utilised to determine the difference in the translations that are detected between the textures. Unless the position of the features can be easily modified, the activation of this must only occur if the boundary between the different surfaces split the image so the trackers are observing different surfaces. This can be determined quite simply using the line equation, which informs the localisation module to not combine the trackers and to only use the one closest to the robot. One other condition to keep in mind is to do with the type of motion that is currently being carried out by the robot. If the robot is undergoing a rotational motion, the motions detected by the trackers should be different, thus can interfere with the depth detection process. It is possible to use the rotational information derived from the other camera to retrospectively correct the motion, but this issue is simplified by restricting the activation to only when translations occur.

Although this breaks the assumption used in the localisation module that the distance to the ground does not change, it is a realistic issue that can be encountered. Therefore, the localisation module must be modified slightly when the landmark is in view to only make use of the one closest to the robot. If the motion of the other side is deemed to be significantly different, the type of different motion should be observed. Since the focus control on the camera cannot be modified with most webcam models, the blurring of the ground texture can cause the feature tracking to be very inconsistent. However, this is also the case when the material changes to one that is difficult to distinguish, such as rubber, which can be common in indoor environments as flooring dividers.

The technique introduced above contains many flaws related to the reliability of the boundaries, detection of difference in the altitude, and issues with blind spots, thus should not be used as a confident indicator of hazardous discontinuity in the surface. It is possible to improve upon the approach with the current set of sensors, such as by using a larger viewing area for the tracking, but should await for a more specialised sensor to be installed before making confident decisions as the safety of the robot is a critical requirement for any mobile robots.



## 11.3 Omnidirectional vision

### 11.3 Omnidirectional vision

The last of the sensors to be considered is the omnidirectional camera that is capable of simultaneously observing the surrounding environment by using a reflective surface to focus the reflected light into the viewing area of the camera. The camera is mounted at the top of the robot looking up into the dome shaped reflective surface, as shown in figure 11.11.

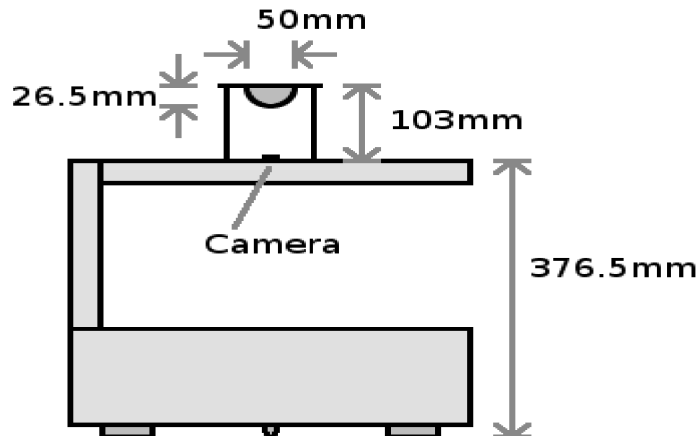


Figure 11.11: Omnidirectional camera placement.

The semi-circle at the top is the reflective dome that allows the simultaneous viewing of the surroundings as the camera looks up towards it.

The placement of the camera has been deliberately made high to reduce the area of the robot within the view, but at the same time, had to consider the stability of the mount and to avoid collisions with objects that were outside the viewing area of the other sensors. With the current configuration, the robot occupies approximately 7% of the image captured by the camera. On top of viewing the robot itself, portions of the image was obstructed from the struts holding the reflective dome in its place. This introduced two solid lines that rendered the portion of the image to be unusable, as well as obstructing the continuity of objects that are behind the struts.

Although many implementations of omnidirectional cameras make use of transparent tubes to hold the camera up, the struts were used due to material availability and to determine the applicability of the wider viewing area before investing in new hardware. For the same reason, the camera that is used is a very cheap model with inferior quality compared to the other cameras used on the robot. The reflective dome that is used is also from recycled hardware that has been coated with reflective paint and differs to the more common cone shaped reflectors.

Another source of the reduction in the usable viewing area is the rectangular 4 by 3 aspect ratio, which expends a large portion of the viewable area on the place holder for the dome. A snapshot from the omnidirectional camera is shown in figure 11.12, where regions that are not useful are highlighted by a red mask. These unusable regions add up to reduces the utility percentage to approximately 35.18%, which can be slightly improved by balancing the distance between the dome and the camera to reduce the outer unusable area and increasing the inner unusable area.

### 11.3 Omnidirectional vision

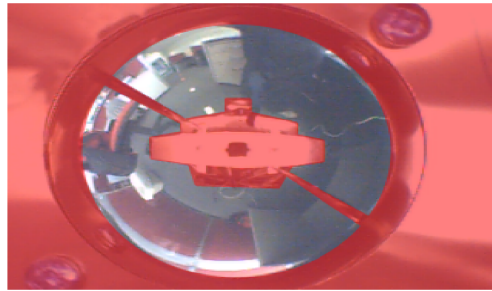


Figure 11.12: Unusable portions of the omnidirectional view. The red mask shows the regions that must be discarded as they do not change or show irregular reflected surfaces.

#### 11.3.1 Analysis

The use of omnidirectional cameras has often been focused on being able to capture the surrounding environment with minimal hardware requirement (Winters et al., 2000). Although the simultaneous snapshot of the entire surroundings can be quite useful, there are additional considerations that require attention to be able to extract and process the captured image. The most apparent characteristics of the image from omnidirectional camera is the warping introduced by the reflective surface, which modifies the compression rate of objects that are at different distance and altitude to the camera. Depending on the shape of the reflective surface that is used, the compression ratio can also change in a non-linear fashion to allow focus on a specific region.

By knowing the exact shape and placement of the reflective surface, it is possible to derive the transformation matrix required to map the captured image to a flat canvas, such that an accurate inter-pixel relationship can be determined (Peters et al., 1996). As the camera and reflective surface configuration is often placed in a rigid formation, a calibration process is often carried out once using a known pattern, such as a grid. Figure 11.13 shows a sample snapshot after the image is unwrapped into a panoramic representation without any vertical adjustments due to the dome surface. The unwrapped image is sometimes blurred afterwards to clean up the artefacts from the non-linear mapping.

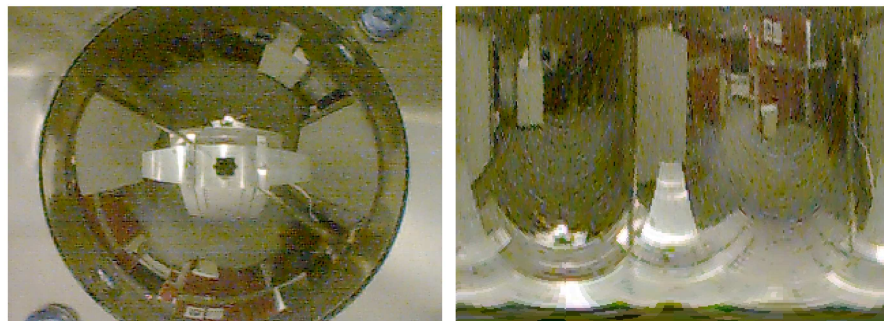


Figure 11.13: Unwrapping of an omnidirectional image. The unwrapped image of the left hand side is shown on the right, which used a simple linear model based on the distance and orientation from the center of the camera.

### 11.3.1 Analysis

When the mapping is carried out, the inconsistent compression rate causes the captured intensities to be stretched or compressed. The mapping process is typically carried out in a per-pixel basis, thus limits the resolution that can be used for the converted image without introducing too much artefacts, especially for interpolated portions of the image. By using a very high resolution to capture the scene at a slow rate, which is often the way the system is used; it is able to produce a detailed image for further analysis. However, when the camera is used to stream a continuous sequence of frames for an up to date state of the scene, the quality of the image suffers significantly due to the reduction in the capturing resolution and the lack of noise reduction that is often carried out by fusing multiple still shots. As a result, the information that can be determined from the image is imprecise and can contain large amount of artefacts from the compression, sensor generated noise, and motion blur. This leads to many of the patterns being suppressed and the precision of any features that are found to be dramatically reduced.

The inaccuracies means the images that are captured from the omnidirectional camera should not be used for precise measurement of the environment's state. However, the images can be used to identify large features, such as lines and colour based segments, to address problems like tracking and localisation (Adorni et al., 2003; Bishay et al., 1994, Gaspar et al., 2000; Menegatti et al., 2004; Shakernia et al., 2003; Vassallo et al., 2002; Winters & Santos-Victor, 1999).

Due to the wrapping of the axes parallel to the ground, any lines that may be present in those directions appear as arcs. These can be difficult to trace, due to the low level of precision and the limited distance the arc spans across. By dewarping the image, some of these arcs can be restored as straight lines, but often results in jagged lines and also consumes valuable processing time. Instead of attempting to make use of these lines, such as for noting when the robot is about to encounter an obstacle that is invisible to the other sensors, vertical lines that are in the scene can be found as lines extending from the middle of the image. Although the precise location of these lines cannot be determined, it is possible to align the robot with the features on the local map using the extra level of constraint provided by the increased viewing area (Brown & Donald, 2000; Cauchois et al., 2003; Franz et al., 1998).

Since the visible range of the local map and the omnidirectional camera differs significantly, correlating the edges that are found and the map can be difficult to achieve accurately. The limited resources in the current implementation means the capture resolution must be set to a very low quality to allow the other devices and algorithms to operate with a higher priority. This makes the identification of lines more difficult, thus the vertical line based correlation is not included in the current implementation. In future implementation, it may be possible to activate the camera at the highest resolution, perhaps when the other sensors are inactive, to capture a more reliable image for better correlation with the other sensor readings. There is also scope for combining the results from the boundary detection algorithm from the side looking camera, as it provides a single point for the boundary to allow for a simple correlation. A prototype of the vertical line detection is illustrated in figure 11.14 as the radial lines, which are derived from a ranked list of Hough transform, are cast onto the local map to illustrate the orientation of vertical lines.

### 11.3.1 Analysis

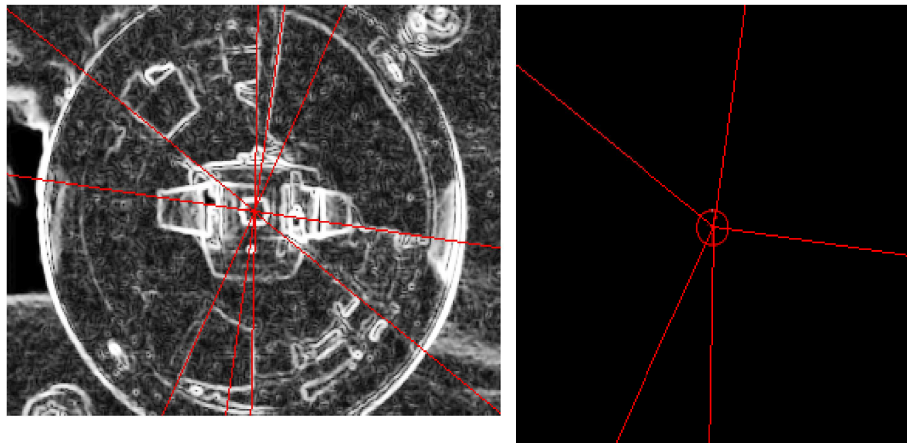


Figure 11.14: Casting of vertical lines from the omnidirectional camera. The lines are selected from the top five lines determined by the Hough transform, which has been cast onto the local map as the right image.

### 11.3.2 Scene changes

The other type of analysis that can be carried out using the images is based on segments, such as grouping those with similar intensities or behaviour (Stocker, 2002). The unique characteristics of the omnidirectional camera is utilised to observe the presence of dynamic objects within the scene, as it is able to track its motion without any physical adjustments. Although the low level of precision and the noise ratio can be problematic, it is possible to observe large objects that appear within the scene, especially if there is distinctness within the intensity levels. By combining this with the availability of the streams of sequential images, an analysis on the change in the scene can be carried out.

The first consideration to make when attempting to observe the dynamic objects amongst static objects is with regards to how they will be detected. Since the motion of objects should be continuous, even with the warping, the object should appear near its previous location in between frames. This means an object in motion will produce a pair of regions when two consecutive frames are compared, where one is the new location of the object and the other is the old location of the object. By making use of a temporal difference filter, it is possible to identify the regions that change, but must take into account that the subtle motions that occur causes very small changes to the intensity which may be below the noise level threshold. Reducing the noise level threshold can dramatically increase the false positives, which increases the number of regions to be processed in the next phase.

Since the process is never intended on being accurate, it is possible to apply a blurring filter to suppress some of the noise. Since the detection of the interesting areas uses a temporal filter, the interpolation is carried out with the spatial neighbours. By using a strong interpolation mask, some of the boundaries that may have existed can be suppressed as a side effect. Similarly, the size of the mask used to interpolate the neighbours is kept small to minimise the de-localisation of the intensity boundaries.

### 11.3.2 Scene changes

The activation of this check can occur at various times depending on what is intended on being observed. Since observing the scene while the robot is in motion will cause the majority of the scene to change, it does not allow a focused analysis on the dynamic object. To avoid mistaking the change in the view as dynamic object when the robot is in motion, it is possible to make use of the information from the localisation module. As noted earlier, this can be slightly problematic if the modules are not synchronised properly, thus requires a buffering of the motions to make sure no robot motion occurs when the change in the image is observed.

An alternative check to distinguish between a dynamic object and the robot motion is to observe the spread of the regions that are observed as having changed. If the robot undergoes a motion in a densely occupied area, there will be change detected after the temporal filter across a wide range of areas within the image. However, if a dynamic object is present and the robot is stationary, the portions that are changing is localised to one region. Although this assumption does not cater for the presence of multiple dynamic objects, such as a crowded room with people, but allows the decoupling of the modules.

The process begins with filtering out the non-relevant portions of the image, which is pre-determined and noted in a lookup table. Since misalignments can occur from the rocking motion, the filter is increased in size by one pixel to make sure irrelevant images are not included. When traversing through the pixels of interest, they are interpolated with the neighbours using a large weighting to encourage the smoothing of the intensities.

$$\text{Intensity}_{x,y} = (4 * \text{Intensity}_{x,y} + \text{Intensity}_{x+1,y} + \text{Intensity}_{x-1,y} + \text{Intensity}_{x,y+1} + \text{Intensity}_{x,y-1}) / 8 \quad (57)$$

Note that the correction of the codec induced artefacts is not carried out for this, as the capture resolution is set too low for the blocks to form. Even if the resolution is increased, the blurring carried out by the above can remove the visible blocks, as the weights that are used is much larger than the block removal algorithm.

The filtered image is then compared with the previous frame to note any changes in the intensity. Due to the lack of a consistent light source and the reflective material that are within the irrelevant portions of the view, the typical intensity level of the view was lower, which promoted more noise. However, since the blurring algorithm is able to suppress the majority of these, the noise level threshold is only increased slightly. Once the pixels that have been noted as having changed are identified, the coordinate points are processed to distinguish the camera motion to dynamic object motion. Note that the use of just the temporal filter without the removal of the non-relevant pixels is not done due to the reflections that appear on the shiny surfaces, which can appear as having changed in colour.

By converting the coordinate point of the pixels to polar coordinates, the accurate orientation of the pixels can be determined. However, since it is only the distribution of the pixels that is of interest, the coordinate points can simply be averaged. If the average coordinate point is located near the middle of the image, the motion can be classified as a change in the robot's pose. Otherwise, the pixels that are involved are noted as being a part of a dynamic object. The threshold used to distinguish this is defined roughly by the circumference of the robot, which is not an accurate way of distinguishing the two types of motions, but is reasonable for this prototype.

### 11.3.3 Cluster

### 11.3.3 Cluster

With the potential pixels for dynamic objects identified, these can then be combined to observe the structure of the dynamic object and also eliminate spurious pixels that are insignificant or were generated by noise. Performing a robust clustering algorithm often requires multiple analysis of the same image, which is often carried out off-line. By specifying constraints based on the expected configuration of the cluster and the domain knowledge about the typical structure of the scene, it is possible to reduce the complexity to be able to operate in real time.

The simplest form of clustering involves a proximity and counter or size based requirement, where a minimal number for the pixels in the group is specified before it can be recognised as a cluster. The apparent size of objects can differ greatly depending on how far away the object is, as well as the actual size of the dynamic object, thus it must be made small. By setting this too small, it can also register the artefacts as dynamic objects, which can hinder the performance and potentially corrupt the state of the map by incorrectly flagging a static object as being dynamic. Since the blurring can potentially spread a single noisy pixel to five pixels, this can be used as the threshold condition to differentiate a noise from a cluster.

An alternative approach to reducing the noise is the use of a temporal blurring before the temporal difference filter is applied. This allows the location of the intensities to remain stationary and enlarges the areas that are noted when motion occurs. Although this allows for a slightly more distinctive regions being shown, there is a small delay introduced in noticing the change. Depending on the requirements of the system, the two types of filters can be interchanged.

When grouping the different pixels, it is possible to make use of the intensity characteristics to add another constraint to the clusters that are formed. This can be used to distinguish the different dynamic objects that may be simultaneously moving near each other. This type of analysis is useful for a more long-term object tracking, which can note and make use of the different motion behaviours to identify the separate objects being involved.

Using an intensity based segmentation algorithm, it is possible to extend the boundary of the dynamic object to include the whole surface. This means the motion behaviour could also be applied to regions that did not initially seem interesting. One of the issues with extending the area of the dynamic object is the lack of confirmation on the connectivity of the pixels other than the similarity between the intensities. This can result in unrelated regions being marked as being a dynamic object.

A related issue with the above is the connectivity between the detected pixels that originate from the same object. Other than the possible gaps between the flagged pixels that are caused by the temporal filter threshold, the dynamic object may be obstructed by either another object within the scene or the robot itself, such as the strut holding the reflective dome. For a small gap, this issue can be accounted for by increasing the search area for the adjacent pixel, but it does not provide a distinction between multiple objects and an obstructed object. For regions where the discontinuity occurs from falling below the threshold, it is possible to observe the low amount of intensity difference at the pixels surrounding the gap to allow the

### 11.3.3 Cluster

connection. If, however, there is a large difference in the change in intensity it can be deemed that the boundary of the object has been reached, or it is being obstructed by a foreground object. Similarly, if the distance between the two pixels is too large, the pixels are not joined as the analysis currently only considers the immediate state of the scene. The identification of the dynamic object is only required in the map, where the cell grids are marked as being static or dynamic, thus the continual tracking and correct grouping is not a critical requirement.

Once the pixels are marked and grouped together, the object's position and orientation can be determined. To define the bounds of the dynamic object, the sector containing the region is used to indicate the direction of the dynamic object. Identifying the bounding sector is a simple matter of converting the pixel's Cartesian coordinates to polar coordinates and selecting the two extreme angular values. It is also possible to identify the strip within the sector which corresponds to where the object may be located by finding the two extreme radius values, which will form a bounding area for the object.

Using the single frame, neither the altitude nor distance to the object can be determined without domain knowledge. This means the depth must either be assumed based on how much of the ground is visible before reaching the object or derived from alternate sensor measurements. Since the ground texture can differ to invalidate the segmentation algorithm and the placement of the dynamic objects does not always need to extend straight up from the ground, the distance measure that is derived can be misleading.

To assist the process of determining where the dynamic object may be located, an approximate value is derived from the radial distance, as shown in figure 11.15. This was determined by placing markers on the ground at known distances away from the robot and determining the position within the image using a capture resolution of 160 x 120 pixels. This value is then combined with the occupancy map to establish where the dynamic object may be.

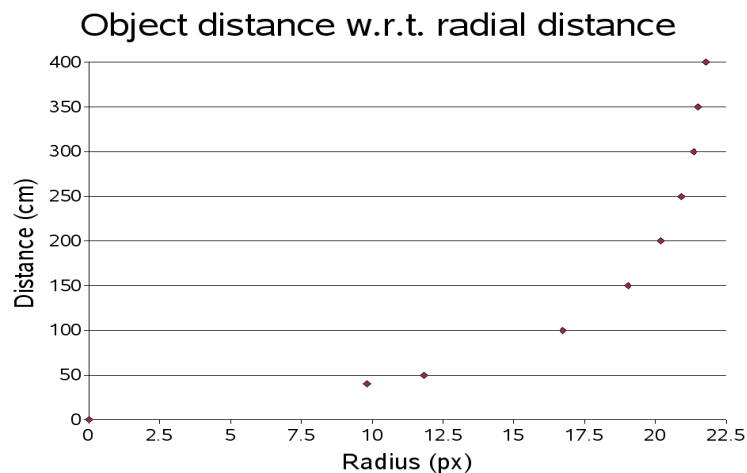


Figure 11.15: Approximate distance to objects with respect to radial distance.

The relationship between the distance to the object and the positions within the image can be determined for flat objects on the ground surface.

### 11.3.3 Cluster

On top of this, the vanishing point can be used to limit the validity of the approximation by specifying a maximum distance for which the assumption is valid. Although the calibration process above points to a distance of approximately 22 pixels from the center, the level of accuracy that can be distinguished diminishes very rapidly, thus should not be used near this point. Since the current size of the local map is limited to a size of approximately 5 by 5 meters, this distance is used to ignore the objects that are detected at a further distance than 21 pixels away from the center. This condition means some dynamic objects that are off the ground will not be registered. However, since the local map only includes obstacles that are viewable by the range finders, this is not a significant issue.

### 11.3.4 Negative carving

Based on the approximate location of the dynamic object, the equivalent regions within the maps can be marked to note that the obstacles that the range finders found can be separated from the map. Since the portions that are flagged are only the boundaries of the dynamic object, this process cannot occur directly. Instead, the focus is placed on marking the cells as containing dynamic objects and modifying the occupancy and vacancy values to reflect the uncertainty in the current location of the object.

Assuming that the dynamic objects that are detected lies above the ground level, the distance that is measured for the object represents the upper bound to the actual distance to the object. Although the sector that is derived confines the possible location of the dynamic object, it should not affect the static objects that lie behind them. To identify the foremost object within the sector, techniques such as ray tracing or incremental arc tracing can be done until occupied cells are encountered.

When tracing an arc for the location of the object, it may be that the dynamic object has recently moved into that area and the occupancy of the cell may not be registered. To account for this, two rays are traced, which are based on the sides of the sector to identify the edge of the object. The pair is used to identify the closest object along the rays which indicates the old position of the object that have now moved.

Based on the motion direction of the dynamic object, the other ray will either be near another occupied cell or will not find an occupied cell. In the latter case, the other ray does not allow for the end point of the dynamic object to be determined, thus modifications to the map can be misleading if multiple cells along the line are modified as containing a dynamic object or their occupancy score reduced. If, however, occupied cells are found for both rays near each other, the region in between the two cells can be modified such that the dynamic attributes are increased and the occupancy scores decreased. This assumes that the dynamic object is visible to the range finders and connected, such that the cells between are occupied cells. Note that the regions near the maximum radial distance are never used as part of this analysis, as they often represent the altitude of the object.

The distribution of the pixels along the minimum value can be tracked by tracing between the bounds of the sector. To simplify the check, a line can be traced instead of allowing dynamic adjustments to the path, as the angle between the bounds are



### 11.3.4 Negative carving

often very small due to the frame rate and the speed of objects. If the dynamic object has an irregular shape, the line may bend or be jagged, thus the thickness of the line is increased to the adjacent pixels like in the landmark detection algorithm. This is illustrated in figure 11.16, where the lower sector in blue successfully identifies the occupied cell, which is shown in green, and traverses between them, which is shown in cyan.

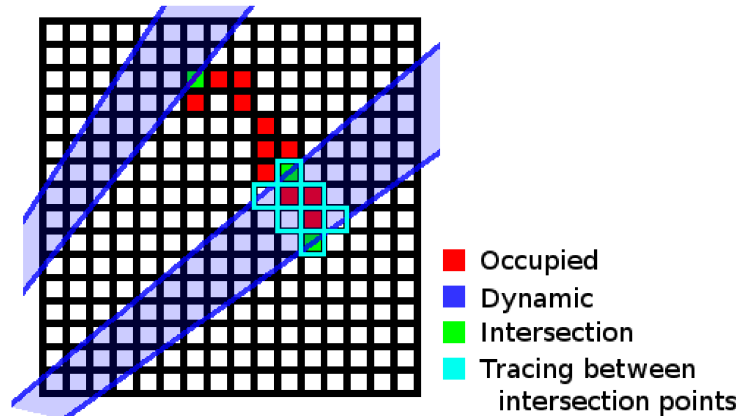


Figure 11.16: Tracing the dynamic object boundary for surface continuity.

The blue regions are the bounds of the temporal filter and clustering that has been cast onto the local map. The green cells are the intersecting points of the blue region and the red region, which are the occupied cells in the local map. The traversal begins and ends at the green cells, as shown by the cyan squares.

If the traced cells successfully reach the other intersected cell, these can be marked as now vacant. This is done by reducing their occupancy value and increasing their vacancy value. At the same time, the angle of the last access is reset, such that the subsequent scan can potentially correct this error if the assumption about its motion is incorrect.

Since the accuracy and the reliability of neither the map nor the omnidirectional image are not high, the intersection point may or may not be reached at the desired location. This is accounted for by narrowing the angle between the bounds of the dynamic object. Since the region in-between should be occupied, the line does not need to be made thicker while finding the intersection cells in the local map. However, when the occupancy is being removed from the region between the intersection points, the surrounding cells are also included and a subsequent range finder measurement is encouraged to observe the obstacle again for an up to date view.

Alternative approaches that were attempted use much simpler techniques of reducing the occupancy of the affected region regardless of the arrangement of the pixels within the boundary. The first approach only makes use of the sides of the sector to form a triangle which extends to the edge of the local map and simply reduces the occupancy of all cells covered. The assignment of the dynamic attribute is carried out in a similar fashion, where all of the cells' attributes is incremented. A slightly modified implementation assigns a different weight to the attributes based on the distance from the robot, where the values are decreased linearly from the base of

### 11.3.4 Negative carving

the robot to zero at the edges of the map. By using the radial bounds, the maximum distance of this can be specified, with the shape being converted into a sector, such as that shown in figure 11.17.

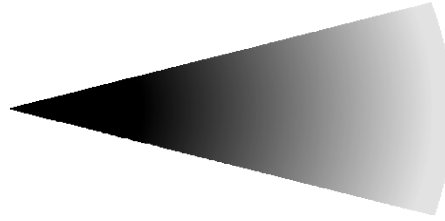


Figure 11.17: Weight distribution for attribute modification after detection of dynamic objects.

The dark colour represents a low weight, while the bright colour represents a higher weight.

Since these approaches make no confirmation with the other sensor readings, the local map is over modified from the large number of cells accessed. Although the narrow sector limits the number of false flagging, relying on multiple observations of the dynamic object for confidence or voting is not effective as their motion may not be continuous.

## 11.4 Connectivity map

The use of the grid map allows the maintenance of detailed and easily accessible attributes, but does not allow for an effective representation of sparsely located attributes, such as high level constructs like landmarks, detached objects, and semantic tags. An example that requires separate representation is the ground texture landmarks that contain the line equation, pose of the robot when the landmark was captured, and the average intensities of the two sides adjacent to the line. These attributes and the sequence in which the landmarks were derived can allow extra information to be derived, such as the path of traversal and connectivity between them.

By linking the landmarks together using a graph structure, it is possible to define the connectivity between the landmarks. If it is simply the traversal between the landmarks that is required to connect them, the connectivity map does not provide useful information, as it does not encourage the use of the existing landmarks during the traversal, nor does it maintain the changes in the motion commands used between the landmarks. Since the ability to travel from one landmark to another is always allowed except when the path is blocked by dynamic objects, this information is redundant and would not be used. Instead, a separate map is introduced to maintain the path of traversal of the robot while capturing the connectivity between the pause points of the robot's traversal. The landmarks are thus left to be used for recalibration of the robot pose.

The current traversal modes for the robot includes two basic types, where one requires constant manual intervention to specify the individual motor commands, while the other makes use of automatically generated motor commands based on the immediate proximity of obstacles surrounding the robot. The orientation of the robot

## 11.4 Connectivity map

is modified based on the most vacant orientation determined from the IR sensor reading. The traversal distance is randomly chosen between 50 cm and 2 m, which can be interrupted when an obstacle reaches too close to the robot. In both modes of traversal, there is a distinct pause between the motor commands, which allows the localisation algorithm to catch up from the buffering of the motion, as well as allocating some time for the dynamic object detection to occur.

Using the distinct steps in motion, the coordinate points, as well as the occupancy of the surroundings, can be used as the nodes for the connectivity map. The nodes of the map can be connected in order of traversal, but the occupancy of the surroundings is used to identify alternate paths that may exist between other nodes.

The measure of the occupancy surrounding the pause point is derived from the smallest IR distance from the robot, which is converted to a circle centered at the robot. The circle represents the amount of space the robot could freely move around in, which is compared with other nodes to determine if one overlaps another. Since the number of nodes does not get too large for the current style of execution, the check is conducted when the new node is created. If an overlap between the circles occurs, the nodes are added to the list of inter-node connections to expand the possible paths between the nodes. If, however, one node completely overlaps another, which is when the distance between the nodes is smaller than the magnitude of the difference in the radius, the smaller node can be eliminated after copying all the connections over to the larger node.

Figure 11.18 illustrates the connectivity nodes superimposed over the global map, where the yellow lines, which were manually added, indicate a connection between the nodes, shown in green. The motion commands were issued manually with distinct pause points to adjust the orientation of the robot. The overlapping and merging of the nodes occurred when the robot was rotated to allow the sonar to scan the surroundings. Note that since the node is constructed using the minimal distance to an obstacle measured by the IR sensor, the nodes indirectly represent an unoccupied region, with some allowance from sensor errors and blind spots. This indicates that a path can be constructed from anywhere within the node to another by moving to the center of the node before and after traversing between different nodes, given that no moving objects enter in the path. This form of path planning allows a quicker formation of paths by re-using previously used paths instead of planning new paths every time the robot travels.

Since the accuracy of the IR sensors is quite low and assumptions are made with regards to validity of the occupancy around the robot, the connectivity map can contain erroneous areas that can lead to incorrect paths being constructed. A more accurate representation of the surroundings can be conducted by spinning the robot by approximately 30 degrees at the pause points to cover the blind spots of the IR sensor, or rotating the sonar sensor, but has been left out for future implementation. An alternative approach is to make use of the occupancy of the local map instead of the IR sensor readings. However, this requires a lengthier radial scan of the map, waiting for the map to be populated with confident information, as well as dealing with the complexity of applying a threshold to distinguish an certain obstacle to a false positive obstacle.

By implementing a more meaningful and coupled traversal commands, it is

## 11.4 Connectivity map

possible to avoid many of the redundant operations when the robot attempts to carry out the specific tasks they are given, such as following a wall or exploring unvisited areas (Burgard et al., 1997; Whaite & Ferrie, 1997). If multiple goals are defined, they must specify priorities such that the traversals do not interfere with each other and that the mapping modules be running parallel.

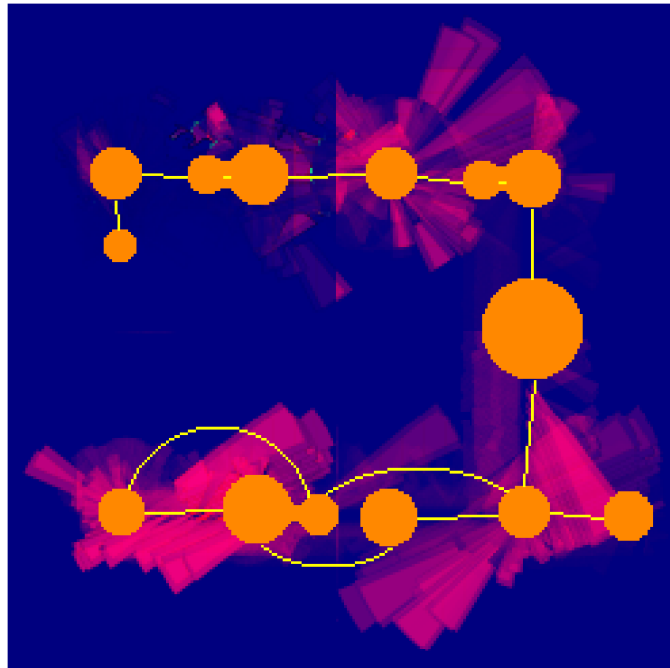


Figure 11.18: Connectivity map superimposed on top of the global map. The pink regions represent vacancy, the green represents occupancy, and the orange spots represent the nodes where the robot was issued a new motion command. The size is determined by the IR sensor reading at the time of the command being issued. The yellow lines joining the orange dots were hand drawn to illustrate the connectivity between these nodes.

## 11.5 Summary

The vast types of different interactions the robot and the sensors can make to the environment allows for any number of algorithms to be implemented to assist the tasks given to the system. Several different techniques and algorithms have been introduced to assist the robot localisation and modelling of the environment. The majority of the algorithms focused on the use of the visual information from the webcams, but also included the use of information that were directly and indirectly derived from other types of sensors.

The grouping of the features were carried out by observing the similarities and relationships between the features derived from chapter 10 to form object boundaries that can be easily translated onto the local map. Since the application focuses on real time processing of the information, more robust segmentation or clustering algorithms were avoided and a more specialised grouping of features were

## 11.5 Summary

implemented. In doing so, some domain knowledge and constraints were introduced to improve the efficiency and effectiveness of the algorithms, such as by reducing the search areas, which resulted in a more consistent pose of boundaries to be found.

The texture based segmentation provides high level surface continuity information, which can be used for isolating different portions of the map and smoothing the boundaries determined by the range finders. The technique that is introduced focus on fast processing using both the hue and luminance scales, which can potentially be improved by merging the results from multiple resolutions or adding shape and template based patterns to classify the group. Although the derived groups are not included in the current implementation, there are scope for this information to be used to improve the accuracy and consistency of the local map.

By re-using the image streams from the cameras pointing downwards, landmarks could be determined based on detecting significant changes to the texture pattern. The landmarks are currently based on straight lines to increase the likelihood of encountering the landmark again while providing for some drifting error correction. The approach includes the quick tests to determine the presence of texture changes as well as considerations for the frequency of the check to reduce unnecessary processing. The landmarks that are stored is currently maintained in a simple list implementation, but should ideally be converted to a more scalable data structure, such as a bucket or quad-tree like structure. This will allow faster access to the most appropriate landmark by using some of the known state of the robot.

The use of the omnidirectional camera has many potential benefits, but is hindered by the resource availability and the small viewing area utilisation that only allow sub-standard image quality. This limitation meant the analysis that were carried out could not be done with much accuracy and provided little benefit to the current system other than the rough measure of identifying the presence of dynamic objects. It was noted that by de-activating the other modules and specifying a higher resolution to capture the scene, the characteristics of the scene was much better perceived. This hints to a potential usage of the sensor, which is by selectively switching between the modules depending on the current state of the robot as long as the overheads in the algorithms remains small and the correct conditions for the switching can be established.

## Chapter 12 – Conclusion

The physical platform provided by the mobile robot has allowed for a multitude of algorithms and techniques to be integrated as part of the modelling process of the environment. The focus has been placed in the real time and simultaneous processing of multiple sensor readings to construct accurate and informative maps. As the project encompasses a significant portion of the entire mobile robot system, the development of the fundamental modules for basic operations defined the structure and aspects to target.

The localisation and mapping processes were carried out using off-the-shelf sensors as both integrated and modular components to allow flexibility in the components that make up the robot depending on the task and resource availability. Many of the proposed algorithms and techniques have been implemented to couple with the product of another module, thus they can be interchanged with ease as long as the fundamental representation of the environment is not drastically modified.

By reducing the amount of input data from the sensors, mainly the capture resolution used by the cameras, the proposed modules are able to be executed concurrently on the mounted laptop. The limited processing capacity means that the precision of the map being constructed is reduced, which can be catered for by temporary disabling some of the modules like the dynamic object finding, reducing the number of feature candidates, not considering the upper half of the side viewing camera for vertical lines, and limiting the operational speed of the robot to reduce the amount of changes in the scene.

### 12.1 Contributions

The components of the mobile robot system that have been developed undertook many incremental improvements by observing the data and constraints provided by the tasks. Many of the resulting implementations are both novel and improved approaches for real time processing, while some are simple implementation of existing approaches with slight differences in the configuration to suit the current system.

#### 12.1.1 Sensor characteristics

The use of the range finders has been accompanied by minor techniques to reduce the inconsistency and improve the speed of translating the sensor readings to a uniform representation. This is done by observing that the measurements are made of natural objects with mostly smooth surfaces.

During the calibration stage of the camera, the sensor's noise characteristic is taken into consideration by observing the fluctuations in the measured intensity reading of a known colour. Although some of the noise characteristics could not be well utilised for a real time system due to resource consumption issues, some of the

### 12.1.1 Sensor characteristics

findings led to generalised algorithms and thresholds that could be defined for reducing the artefacts that are introduced into the image stream.

The radial warping effect seen in many older cameras have been dealt with by simply cropping the image, while the codec induced grid like noise has been dealt with a custom interpolation filter to blend the borders of the blocks. The filters provide an important role of suppressing artificial trends from appearing and quickly removing regions that are blurry.

### 12.1.2 Local localisation

The proposed localisation technique is based on correlating the ground textures between frames to accumulate the motions observed. This is achieved by selecting the most outlier scored feature and tracking its motion in the subsequent frame. The search strategy introduces a radial scan pattern to quickly establish the most appropriate correlation based on previous and current motions.

The synchronisation between multiple trackers on multiple devices required sub-pixel motions to be derived, which was achieved through a weight based blending technique between the detected motions. This also introduced a slight delay in the registration of the motion, but drastically reduced the precision errors between the feature tracking.

The introduction of a hybrid motion model to translate the feature motions to the robot motion allowed both a smooth and unexpected motions to be accounted for. The algorithm switches between different levels of constraints on the motion characteristics based on the type of motion that is observed to reflect the validation of the assumptions used when including the constraint.

### 12.1.3 Map construction

The core component to the local map is based on an occupancy grid map, which is used to store multiple attributes that store how the grid cells were modified, allowing repeated interactions to the map to change depending on the relevance. The most significant attribute is the sensor orientation value to ignore repeated scans from the same or similar orientation as the sensor behaviour can differ greatly when the perspectives change. This allows the sampling rate to be independent of the attributes that are stored.

The accesses to the map is carried out much like a raster image to allow simple inter-cell interactions and well established algorithms like anti-aliasing and compression. The superimposition of the sensor scans are carried out using an area based anti-aliasing algorithm with various enhancements on the speed of both line and arc drawing. Some approximations are introduced during this process, but the accuracy remains high enough for representation purposes.

Due to the isolated calculations and the constraints defined during the above process, optimisation approaches are suggested using fixed point arithmetic operations during the sensor superimposition stage. Although not significant, the approach showed some improvements in the performance, which suggests applicability to other areas with arithmetically intensive operations and known data

### 12.1.3 Map construction

ranges. The fast processing allows the rapid sampling from the sensors to obtain a more continuous and up to date measure of the surroundings.

The simultaneous use of maps of multiple scales is achieved through periodic synchronisation from the local map to the global map. Several strategies and considerations have been suggested to efficiently update portions of the map and to translate the various attributes that are maintained. The maintenance of the attributes also include the considerations when the robot explores more than the expected amount of area. Several strategies are introduced to handle the changes to the local and global maps, such as compression and states of new areas that are introduced.

### 12.1.4 High level visual features

The selection of visual features has been deliberately made with domain knowledge constraints to assist in the fast processing of object boundaries that can be related to the attributes found on the maps. The criterion for the features has been configured to focus on vertical boundaries with consistent colour, which are treated as the meeting point of foreground and background surfaces. This allows the background to change without affecting the validity of the feature.

Several filters, including temporal, density, and ranking, are included to cut down the number of feature candidates. This reduction allows more complex analyses to be carried out later using a smaller set of data. Although this issue can be avoided with reduction in the number of simultaneous processes, the filters were necessary to allow the other modules to be executed simultaneously on the mobile robot.

The grouping of the boundary features using proximity constraints allowed for the feature pose to be converged more rapidly. This reduced many of the precision errors that are introduced when tracking visual features on a camera, as the number of samples used to triangulate the pose was dramatically increased compared to tracking a single feature.

By re-using the image streams from the ground texture tracking cameras, the strategies in capturing a long term landmark has been introduced. The approach focuses on straight line boundaries of texture pattern changes which allows for the reduction in the search area and criteria. The frequency of the landmark detection is also considered to allow the robot to move to another location before searching for a different landmark.

Using the omnidirectional camera, a dynamic object detection algorithm has been included with the appropriate map modification algorithm to mark and reset the range finder readings. The algorithm involves a temporal filter and a distribution based distinction to distinguish the difference between a robot motion and a dynamic object motion. The cluster of pixels are grouped together to identify the sector around the robot, which is then superimposed over the local map to make the modification.

### 12.1.5 Connectivity map

To allow for efficient path finding between previously visited portions of the environment, a graph based connectivity map is introduced to connect between the point of motion commands issued to the robot. The simple implementation includes



### 12.1.5 Connectivity map

basic vacancy check around the node, as well as strategies for the merging of multiple nodes to indicate direct and indirect paths without reverting back to the local map.

## 12.2 Future work

The enormous range of scope for the mobile robot project means the future direction of the project can vary significantly depending on the specific interest of those involved. However, there are several fundamental components that should be enhanced or improved in the future.

A fundamental component that is missing in the current system is a clear definition of the overall and instantaneous goal which drive the operation of the robot. Although an arbitrary goal is defined by the individual user of the robot, there is no framework for defining tasks and decision making process in the current system. This includes navigational (Ahuactzin et al., 1991; Arkin, 1987; Bennewitz, 2004; Buffa et al., 1993; Fiorini & Shiller, 1995; Floreano & Mondada, 1996; Kim, 2004; Latombe, 1999; Miura et al., 1999; Taylor & Kriegman, 1998; Thorpe, 1984; Zelek, 1995; Zimmer, 1996) and specific sensor usage to focus its attention on specific points of interest (Huntsberger, 2001). The inclusion of this type of module will allow simple transition between different tasks which make use of the base operations that have been developed so far.

Another key addition that is required is the inclusion of more hardware for control, interactions, and sensing of the environment. The inclusion of additional sensors will allow more sophisticated interactions, such as orientation from compass sensors, while the upgrading of existing sensors will allow more precise measurements to be made. As the project will continue to be incrementally developed, this aspect will naturally be targeted as the tasks will define the sensors that are required.

In terms of improving the approaches implemented so far, there is a wide range of areas that could be explored, which include:

- Adaptive and automated calibration processes to suit each environment (Tsai, 1987; Quan, 1996).
- The use of parallel processors for the feature analysis (Horn, 1988).
- Using a single camera with mirrors for the localisation to increase the tracker distance since the majority of the captured image is wasted.
- Converting the map into a 3D representation, as well as deriving higher level understanding of the environment (Leonard & Durrant-Whyte, 1992, Hemayed et al., 1997; Kang & Szeliski, 1997).
- Including set of scripted commands for handling repeated navigational tasks.
- Smoother motion transitions around obstacles (Borenstein & Koren, 1989; Lengyel et al., 1990).
- Introducing team work between multiple robots to solve the task (Fox et al., 2000; Rekleitis et al., 1997; Thrun, 2001).

## 12.2 Future work

- A more objective integration process between the multiple sensor readings.

With the improvements in the image quality of the omnidirectional camera, there is scope for more integration with other modules, perhaps even to the extent of replacing the other devices or modules, such as the localisation module (Francis et al., 2006; Spacek & Burbridge, 2007). This would involve balancing of the processing load between the other modules and the development of alternate techniques to compliment or duplicate the behaviour of the other devices.

## Bibliography

- Aboshosha, A & Zell, A 2003, 'Robust Mapping and Path Planning for Indoor Robots based on Sensor Integration of Sonar and a 2D Laser Ranger Finder', *In Proc. IEEE 7th International Conference on Intelligent Engineering Systems*, pp. 4-6.
- Adams, JE 1997, 'Design of practical color filter array interpolation algorithms for digital cameras', *in Proc. SPIE*, vol. 3028, pp. 117-125.
- Adorni, G, Cagnoni, S, Mordonini, M & Sgorbissa, A 2003, 'Omnidirectional Stereo Systems for Robot Navigation', *In Proc. of IEEE Workshop on Omnidirectional Vision and Camera Networks*, pp. 79-89.
- Ahuactzin, JM, Talbi, EG, Bessiere, P & Mazer, E 1991, 'Using Genetic Algorithms for Robot Motion Planning', *Workshop on Geometric Reasoning for Perception and Action*, vol. 708, pp. 84-93.
- Alexander, JC & Maddocks, JH 1989, 'On the kinematics of wheeled mobile robots', *The International Journal of Robotics Research*, vol. 8, no. 5, pp. 15-27.
- Araujo, EG & Grupen, RA 1997, 'Feature extraction for autonomous navigation', Technical document, University of Massachusetts.
- Araujo, EG & Grupen, RA 1998, 'Feature detection and identification using a sonar-array', *IEEE International conference on Robotics and Automation*, no. 2, pp. 1584-1589.
- Ardaiz, M, Astigarraga, A, Lazkano, E, Sierra, E & Martunez-Otzeta, JM 2005, 'Dynamic pan, tilt and zoom adjustment for perception triggered response', *CAEPIA--TTIA*, vol. 2, pp. 151-160.
- Arkin, RC 1987, 'Motor Schema-Based Mobile Robot Navigation', *In Proc. IEEE International Conference on Robotics and Automation*, vol. 4, pp. 264-271.
- Arnold, B 2004, 'Mobile Robot for The Development of Real/Virtual world Model', Thesis, Flinders University.
- Astigarraga, A, Lazkano, E, Sierra, B & Rano, I 2004, 'Active landmark perception', *In Proceedings of the 10th IEEE International Conference on Methods and Models in Automation and Robotics*, pp. 955-960.
- Bahl, P & Padmanabhan, VN 2000, 'RADAR: An In-Building RF-based User Location and Tracking System', *INFOCOM*, vol. 2, pp. 775-584.
- Bailey, T 2002, , *Australian Centre for Field Robotics*, Sydney.
- Baker, S & Nayar, SK 1998, 'A theory of catadioptric image formation', *In Proc. 6th International Conference on Computer Vision*, vol. 30, pp. 35-42.
- Balmelli, L, Ayer, S & Vetterli, M 1998, 'Efficient algorithms for embedded rendering of terrain models', *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 914-918.

## Bibliography

- Balmelli, L, Kovacevic, J & Vetterli, M 1999, 'Quadrees for Embedded Surface Visualization: Constraints and Efficient Data Structures', *In Proc. of IEEE International Conference on Image Processing*, vol. 2, pp. 487-491.
- Bank, D 2002, 'An Error Detection Model for Ultrasonic Sensor Evaluation on Autonomous Mobile Systems', *In Proc. of the 11th IEEE Int. Workshop on Robot and Human interactive Communication*, pp. 288-293.
- Barron, JL, Fleet, DJ, Beauchemin, SS & Burkitt, TA 1992, 'Performance of optical flow techniques', *CVPR*, .
- Basri, R & Jacobs, D 2000, 'Lambertian Reflectance and Linear Subspaces', Technical document, Weizmann Institute of Science.
- Basri, R & Weinshall, D 1993, 'Distance Metric Between 3D models and 2D Images for Recognition and Classification', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 220-225.
- Basye, K, Dean, T & Vitter, JS 1989, 'Coping With Uncertainty in Map Learning', *Machine Learning*, vol. 29, no. 1, pp. 65-88.
- Beauchemin, SS & Barron, JL 1995, 'The computation of optical flow', *ACM Computing Surveys*, vol. 27, no. 3, pp. 433-467.
- Ben-Arie, J & Wang, Z 1997, 'Pictorial recognition using affine-invariant spectral signatures', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 34-39.
- Bennewitz, M 2004, 'Mobile Robot Navigation in Dynamic Environments', Thesis, University of Freiburg.
- Bennewitz, M, Stachniss, C, Burgard, W & Behnke, S 2006, 'Metric Localization with Scale-Invariant Visual Features using a Single Perspective Camera', *European Robotics Symposium*, pp. 195-208.
- Betge-Brezetz, S, Chatila, R & Devy, M 1995, 'Object-based Modelling and Localization in Natural Environments', *In Proc. IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2920-2927.
- Betke, M & Gurdits, L 1997, 'Mobile Robot Localization Using Landmarks', *In Proc. IEEE Robotics and Automation*, vol. 13, no. 2, pp. 251-263.
- Bigas, M, Cabruja, E, Forest, J & Salvi, J 2005, 'Review of CMOS image sensors', *Microelectronics Journal*, vol. 37, pp. 433-451.
- Bishay, M, Peters, RA II, Kawamura, K 1994, 'Object detection in indoor scenes using log-polar mapping', *IEEE Conference on Robotics and Automation*, no. 1, pp. 775-780.
- Black, MJ & Anandan, P 1991, 'Robust dynamic motion estimation over time', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 296-302.
- Blaer, P & Allen, PK 2005, 'A Hybrid Approach to Topological Mobile Robot Localization', Technical document, Columbia University.
- Bleyer, M & Gelautz, M 2004, 'A layered stereo algorithm using image segmentation and global visibility constraints', *ISPRS Journal of Photogrammetry and Remote*

## Bibliography

- Sensing*, vol. 59, no. 3, pp. 128-150.
- Bloch, AM, Krishnaprasad, PS, Marsden, JE, Murray, RM & Holmes, CR 1996, 'Nonholonomic mechanical systems with symmetry', *Arch. Rational Mech. Anal. Archive for Rational Mechanics and Analysis*, vol. 136, no. 1, pp. 21-99.
- Bologiannis, S, Garrod, K & Kennedy, D 2003, 'Intelligent Robot System', Thesis, Flinders University.
- Borenstein, J 1989, 'Real-time Obstacle Avoidance for Fast Mobile Robots', *In Proc. IEEE Transaction on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179-1187.
- Borenstein, J 1993, 'Multi-layered Control of a Four-Degree-of-Freedom Mobile Robot With Compliant Linkage', *In Proc. IEEE International Conference on Robotics and Automation*, no. 3, pp. 7-12.
- Borenstein, J 1995, 'Control and kinematic design of multi-degree-of-freedom mobile robots with compliant linkage', *IEEE Transactions on Robotics and Automation*, vol. 11, no. 1, pp. 21-35.
- Borenstein, J & Koren, Y 1989, 'Real-time Obstacle Avoidance for Fast Mobile Robots', *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179-1187.
- Borenstein, J & Koren, Y 1995, 'Error Eliminating Rapid Ultrasonic Firing for Mobile Robot Obstacle Avoidance', *IEEE Transactions on Robotics and Automation*, vol. 11, no. 1, pp. 132-138.
- Borenstein, J, Everett, HR, Feng, L & Wehe, D 1997, 'Mobile Robot Positioning - Sensors and Techniques', *Journal of Robotic Systems, Special Issue on Mobile Robots*, vol. 14, pp. 231-249.
- Borghi, G & Tosolini, L 2007, 'Real-time Obstacle Avoidance in Socar-based Mobile Robot Navigation Using Harmonic Potential Functions', Technical document, Politecnico di Milano.
- Borzenko, O, Xu, W, Obsniuk, M, Chopra, A, Jasiobedzki, P, Jenkin, M & Lesperance, Y 2006, 'Lights and Camera: Intelligently controlled multi-channel pose estimation system', *IEEE International Conference on Computer Vision Systems*, pp. 42-49.
- Boudihir, ME, Nourine, R & Ziou, D 1998, 'Visual Guidance of Autonomous Vehicle Based on Fuzzy Perception', *IEEE International Conference on Intelligent Vehicles*, pp. 23-28.
- Bouguet, J-Y & Perona, P 1995, 'Visual navigation using a single camera', *In Proc. Fifth International Conference on Computer Vision*, pp. 645-652.
- Bourque, E & Dudek, G 1998, 'Automated Image-Based Mapping', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 61-70.
- Bourque, E, Dudek, G & Ciaravola, P 1998, 'Robotic Sightseeing - A Method for Automatically Creating Virtual Environments', *In Proc. IEEE International Conference on Robotics and Automation*, no. 4, pp. 3186-3191.
- Bresenham, JE 1965, 'Algorithm for computer control of digital plotter', *IBM Syst. J.*,

## Bibliography

pp. 25-30.

- Bresenham, JE 1977, 'A linear algorithm for incremental digital display of circular arcs', *Communications of ACM*, vol. 20, no. 2, pp. 750-752.
- Bretzner, L & Lindeberg, T 1996, 'Feature Tracking with Automatic Selection of Spatial Scales', Technical document, Royal Institute of Technology.
- Brooks, RA 1991, 'Intelligence without representation', *Artificial Intelligence*, vol. 47, pp. 139-159.
- Brown, M & Lowe, DG 2003, 'Recognising Panoramas', *Proceedings of the Ninth IEEE International Conference on Computer Vision*, pp. 1218-1225.
- Brown, RG & Donald, BR 2000, 'Mobile Robot Self-Localization without Explicit Landmarks', *Algorithmica*, vol. 26; part 3/4, pp. 515-559.
- Buffa, M, Faugeras, OD & Zhang, Z 1993, 'A stereovision-based navigation system for a mobile robot', Technical document, Institut National de Recherche en Informatique et en Automatique.
- Buhmann, J, Burgard, W, Cremers, AB, Fox, D, Hofmann, T, Schneider, F, Strikos, J & Thrun, S 1995, 'The Mobile Robot Rhino', *AI Magazine*, vol. 16, no. 2, pp. 31-38.
- Bulata, H & Devy, M 1996, 'Incremental Construction of a Landmark-based and Topological Model of Indoor Environments by a Mobile Robot', *In Proc. IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1054-1060.
- Bulusu, N, Heidemann, J & Estrin D 2001, 'Adaptive Beacon Placement', *International Conference on Distributed Computing Systems*, vol. 21, pp. 489-498.
- Burgard, W, Cremers, AB, Fox, D, Ahnel, DH, Lakemeyer, G, Schulz, D, Steiner, W & Thrun, S 1998, 'The interactive museum tour-guide robot', *In Proc. National Conference on Artificial Intelligence*, no. 15, pp. 11-18.
- Burgard, W, Fox, D & Thrun, S 1997, 'Active mobile robot localization', *In Proceedings of International Joint Conference on Artificial Intelligence*, vol. 15, no. 2, pp. 1346-1352.
- Burgard, W, Fox, D, Hennig, D & Schmidt, T 1996, 'Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids', *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, vol. 2, no. 13, pp. 896-901.
- Burgard, W, Fox, D, Jans, H, Matenar, C & Thrun, S 1991, 'Histogramic in-motion mapping for mobile robot obstacle avoidance', *IEEE Journal of Robotics and Automation*, vol. 7, no. 4, pp. 535-539.
- Burgard, W, Fox, D, Jans, H, Matenar, C & Thrun, S 1999, 'Sonar-based mapping with mobile robots using EM', *International Workshop the Conference on Machine Learning*, pp. 67-76.
- Cafforio, C, Sciascio, ED, Guaragnella, C & Piscitelli, G 1997, 'A simple and effective edge detector', *Lecture Notes in Computer Science*, Springer-Verlag
- Cahut, L, Valavanis, KP & Delic, H 1998, 'Sonar Resolution-Based Environment

## Bibliography

- Mapping', *In Proc. of 1998 IEEE Intl. Conf. on Robotics and Automation*, no. 3, pp. 2541-2547.
- Camus, T & Bulthoff, HH 1995, 'Real-Time Optical Flow Extended in Time', Technical document, Max-Planck-Institut fur biologische Kybernetik.
- Castellanos, JA, Neira, J & Tardos, JD 2001, 'Multisensor Fusion for Simultaneous Localization and Map Building', *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 908-914.
- Cauchois, C, Brassart, E, Delahoche, L & Clerentin, A 2003, '3D localization with conical vision', *In Proc. Computer Vision and Pattern Recognition Workshop*, vol. 7, pp. 81-86.
- Chakarov, D 2006, 'Kinematics Model of Nonholonomic Wheeled Mobile Robots for Mobile Manipulation Tasks', *5th Baltic - Bulgarian Conference on Bionics and Prosthetics*, pp. 59-61.
- Chen, DZ, Szczerba, RJ & Uhan, JJ 1995, 'Using Framed-Octrees to Find Conditional Shortest Paths in an Unknown 3-D Environment', Technical document, University of Notre Dame.
- Chin, RT & Dyer, CR 1986, 'Model-Based Recognition in Robot Vision', *ACM Computing Surveys*, vol. 18, no. 1, pp. 67-108.
- Chong, KS & Kleeman, L 1999, 'Mobile Robot Map Building from an Advanced Sonar Array and Accurate Odometry', *International Journal of Robotics Research*, vol. 18, no. 1, pp. 20-36.
- Chum, O, Pajdla, T & Strum, P 2005, 'The geometric error for homographies', *Computer Vision and Image Understanding*, vol. 97, no. 1, pp. 86-102.
- Clark, JJ, Ferrier, NJ 1992, 'Attentive Visual servoing', *Active Vision*, pp. 137-154.
- Clark, S & Whyte, HD 1997, 'The Design of a High Performance MMW Radar System for Autonomous Land Vehicle Navigation', *Proc. Int. Conf. Field and Service*, pp. 292-299.
- Clemente, LA, Davison, AJ, Reid, ID, Neira, J & Tardos, JD 2007, 'Mapping Large Loops with a Single Hand-Held Camera', *Robotics: Science and Systems*, .
- Coombs, D & Brown, C 1993, 'Real-Time Binocular Smooth Pursuit', *International Journal of Computer Vision*, vol. 11, no. 2, pp. 147-164.
- Corke, PI 1994, 'High-Performance Visual Closed-Loop Robot Control', Thesis, University of Melbourne.
- Crowley, JL 1989, 'World modeling and position estimation for a mobile robot using ultrasonic ranging', *IEEE Conference on Robotics and Automation*, vol. 3, pp. 1574-1579.
- Crowley, JL 1995, 'Mathematical foundations of navigation and perception for an autonomous mobile robot', *Reasoning with Uncertainty in Robotics*, pp. 9-51.
- Crowley, JL & Reignier, P 1992, 'Asynchronous Control of Rotation and Translation for a Robot Vehicle', *Journal of Robotics and Autonomous Systems*, *Journal of Robotics and Autonomous Systems*, vol. 10, no. 4, pp. 243-251.

## Bibliography

- Crowley, JL, Wallner, F & Schiele, B 1998, 'Position estimation using principal components of range data', *In Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3121-3128.
- Csorba, M 1997, 'Simultaneous Localisation and Map Building', Thesis, University of Oxford.
- Cumani, Aldo 1989, 'Edge detection in multispectral images', Technical document, Istituto Elettrotecnico Nazionale "Galileo Ferraris".
- Davison, A & Murray, DW 1998, 'Mobile robot localisation using active vision', *In Proceedings of the 5th European Conference on Computer Vision*, vol. 2, issue 1407, pp. 809-825.
- Davison, AJ 1998, 'Mobile Robot Navigation using Active Vision', Thesis, University of Oxford.
- Davison, AJ 2003, 'Real-Time Simultaneous Localisation and Mapping with a Single Camera', *In Proc. Ninth IEEE International Conference on Computer Vision*, pp. 1403-1410.
- Davison, AJ & Kita, N 2001, '3D Simultaneous Localisation and Map-Building Using Active Vision for a Robot Moving on Undulating Terrain', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I-384-I-391.
- Davison, AJ, Reid, ID, Molton, ND & Stasse, O 2007, 'MonoSLAM: Real-Time Single Camera SLAM', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052-1067.
- Deans, M & Herbert, M 2000, 'Experimental Comparison of Techniques for Localization and Mapping Using a Bearing-Only Sensor', *Lecture Notes in Control and Information Sciences*, Springer-Verlag
- Debevec, PE 1996, 'Modeling and Rendering Architecture from Photographs', Thesis, University of California.
- Del Bimbo, A & Santini, S 1994, 'Motion analysis', *Human and Machine Vision: Analogies and Divergences*, .
- Dellaert F, Fox, D, Wolfram, B & Thrun, S 1999, 'Monte Carlo Localization for Mobile Robots', *In Proc. of the IEEE International Conference on Robotics*, no. 2, pp. 1322-1328.
- Diosi, A & Kleeman, L 2004, 'Advanced Sonar and Laser Range Finder Fusion for Simultaneous Localization and Mapping', *In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1854-1859.
- Draper, BA, Hanson, AR & Riseman, EM 1996, 'Knowledge-Directed Vision: Control, Learning, and Integration', *Proceedings of the IEEE*, vol. 83, no. 11, pp. 1625-1637.
- Drumheller, M 1987, 'Mobile robot localization using sonar', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, pp. 325-332.
- Duckett, T & Nehmzow, U 1998, 'Mobile robot self-localisation and measurement of performance in middle-scale environments', *Robotics and Autonomous Systems*,



## Bibliography

vol. 24, no. 1/2. pp. 57-70.

- Duckett, T & Nehmzow, U 1999, 'Exploration of unknown environments using a compass, topological map and neural network', *In Proceedings of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automatio*, pp. 312-317.
- Duckett, T & Safflotti, A 2000, 'Building globally consistent gridmaps from topologies', *In Proc. 6th International IFAC Symposium on Robot Control*, .
- Dudek, G & Jenkin, M 2000, *Computational Principles of Mobile Robotics*, Cambridge University Press, New York.
- Dudek, G & Jugessur, D 2000, 'Robust place recognition using local appearance based methods', *In IEEE Intl. Conf. on Robotics and Automation*, vol. 2, pp. 1030-1035.
- Dudek, G & MacKenzie, P 1993, 'Model-Based Map Construction for Robot Localization', *In Proc. Vision Interface*, pp. 97-102.
- Dudek, G, Freedman, P & Rekleitis, IM 1996, 'Just-in-time sensing: efficiently combining sonar and laser range data for exploring unknown worlds', *In Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 667-672.
- Dudek, G, Jenkin, M, Milios, E & Wilkes, D 1997, 'Map Validation and Self-location in a Graph-like World', *Robotics and autonomous systems*, vol. 22, no. 2, pp. 159-178.
- Dudek, G, Jenkins, M, Milios, E & Wilkes, D 1992, 'Reflections on modelling a sonar range sensor', Technical document, McGill University.
- Dudek, G, Romanik, K & Whitesides, S 1995, 'Localizing a robot with minimum travel', *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, vol. 27, no. 2, pp. 437-446.
- Eberly, D 1999, 'Integer-based Algorithm for Drawing Ellipses', Technical document, Geometric Tools, Inc..
- Eklund, MW, Ravichandran, G, Trivedi, MM & Marapane, SB 1994, 'Real-Time Visual Tracking Using Correlation Techniques', *In Proc. Second IEEE Workshop on Applications of Computer Vision*, pp. 256-263.
- Eklundh, J, Nordlund, P & Uhlin, T 1996, 'Issues in active vision: Attention and cue integration/selection', *Proc. British Machine Vision Conference*, pp. 1-12.
- Elgazzar, S, Liscano, R, Blais, F & Miles, A 1997, '3-D Data Acquisition for Indoor Environment Modeling Using a Compact Active Range Sensor', *In Proc. IEEE Instrumentation and Measurement Technology Conference*, vol. 1, pp. 586-592.
- Eliazar, A & Parr, R 2003, 'DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks', *In Proc. 18th International Joint Conference on Artificial Intelligence*, pp. 1135-1142.
- Faugeras, O & Robert, L 1993, 'What can two images tell us about a third one?', Technical document, Institut National de Recherche en Informatique et Automatique.

## Bibliography

- Faugeras, OD, Luong, QT & Maybank, SJ 1992, 'Camera Self-Calibration: Theory and Experiments', *European Conference on Computer Vision*, vol. 92, pp. 321-334.
- Feder, HJS, Leonard, JJ & Smith, CM 1999, 'Adaptive Mobile Robot Navigation and Mapping', *International Journal of Robotics Research, Special Issue on Field and Service Robotics*, .
- Feng, D, Friedman, MB & Krogh, BH 1989, 'The servo-control system for an omnidirectional mobile robot', *In Proc. IEEE International Conference on Robotics and Automation*, pp. 1566-1571.
- Feng, L, Borenstein, J & Wehe, D 1996, 'A Completely Wireless Development System for Mobile Robots', *In Proc. ISRAM Conference*, pp. 571-576.
- Fiorini, P & Shiller, Z 1995, 'Robot motion planning in dynamic environments', *International Symposium of Robotic Research*, .
- Fleet, DJ & Langley, K 1995, 'Recursive filters for optical flow', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 1, pp. 61-67.
- Floreano, D & Mondada, F 1996, 'Evolution of Homing Navigation in a Real Mobile Robot', *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 26, no. 3, pp. 396-407.
- Foessel-Bunting, A 2000, 'Radar Sensor Model for Three-Dimensional Map Building', *Proc. SPIE, Mobile Robots XV and Telem manipulator and Telepresence Technologies VII*, vol. 4195, pp. 127-138.
- Fonseca, A 2007, 'Computer Vision Image Processing and Mapping for Mobile Robot', Thesis, Flinders University.
- Ford, JL 1998, 'Color Theory Tutorial by Worqx', *Color Theory Tutorial*, <http://www.worqx.com/color/>.
- Fox, D, Burgard, W, Kruppa, H & Thrun, S 2000, 'A Probabilistic Approach to Collaborative Multi-Robot Localization', *Autonomous Robots*, vol. 8, no. 3, pp. 325-344.
- Fox, D, Burgard, W, Thrun S & Cremers, AB 1998, 'Position Estimation for Mobile Robots in Dynamic Environments', *Proceedings of the National Conference on Artificial Intelligence*, no. 15, pp. 983-988.
- Fox, D, Thrun, S, Burgard, W & Dellaert, F 2001, 'Particle Filters for Mobile Robot Localization', *Sequential Monte Carlo Methods in Practice*, .
- Francis, G, Spacek, L & Park, W 2006, 'Linux Robot with Omnidirectional Vision', *In Proc. Towards Autonomous Robotic Systems*, .
- Franz, MO, Scholkopf, B, Mallot, HA & Bulthoff, HH 1998, 'Where did I take that snapshot? Scene-based homing by image matching', *Biological Cybernetics*, vol. 79, no. 3, pp. 191-202.
- Gall, DL 1991, 'MPEG: A video compression standard for multimedia applications', *Communications of the ACM*, vol. 34, no. 4, pp. 46-58.
- Garnier, P, Novales, C, Pallard, D & Baille, G 1995, 'Autonomy for electric cars', *Proc. of the Electric Vehicle Technology Conf*, vol. 2, pp. 24-33.

## Bibliography

- Gaspar, J, Winters, N & Santos-Victor, J 2000, 'Vision-based navigation and environmental representations with an omnidirectional camera', *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 890-898.
- Geusebroek, J-M, Dev, A, van den Boomgaard, R, Smeulders, AWM, Cornelissen, F & Geerts, H 1999, 'Color Invariant Edge Detection', *Lecture notes in computer science*, vol. 1682, pp. 459-464.
- Ghahramani, Z 1998, 'Learning Dynamic Bayesian Networks', *Lecture Notes in Computer Science*, vol. 1387, pp. 168-197.
- Ghidary, SS, Tani, T, Takamori, T & Hattori, M 1999, 'A new Home Robot Positioning System (HRPS) using IR switched multi ultrasonic sensors', *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, pp. IV-737-IV-741.
- Goel, P & Sukhatme, GS 2000, 'Sonar-based Feature Recognition and Robot Navigation Using a Neural Network', *In Proc. Intelligent Robots and Systems*, vol. 1, pp. 109-114.
- Grabowski, R, Khosla, P & Choset, H 2003, 'An Enhanced Occupancy Map for Exploration via Pose Separation', *In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1.
- Graves, S, Cision, L & Wise, JD 1992, 'A modular software system for distributed telerobotics', *In IEEE International Conference on Robotics and Automation*, pp. 2783-2785.
- Guibas, LJ, Motwani R & Raghavan, P 1995, 'The robot localization problem', *Proc. 1st Workshop on Algorithmic Foundations of Robotics*, pp. 269-282.
- Guivant, JE 2002, 'Efficient Simultaneous Localization and Mapping in Large Environments', Thesis, University of Sydney.
- Guttman, A 1984, 'R-trees: A Dynamic Index Structure for Spatial Searching', *ACM Sigmod Record*, vol. 14, no. 2, pp. 47-57.
- Halperin, D, Kavraki, L & Latombe, J-C 2004, 'Robotics', *Handbook of Discrete and Computational Geometry*, pp. 1065-1094.
- Harris, C, & Stephens, M 1988, 'A combined corner and edge detector', *In Proc. Alvey Vision Conference*, vol. 15, pp. 147-151.
- Harris, LR & Jenkin, M 2001, *Vision and Attention*, Springer Verlag, London.
- Heckbert, PS 1986, 'Survey of texture mapping', *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 56-67.
- Hemayed, E, Sandbek, A, Wassal, A & Farag, A 1997, 'Investigation of stereo-based 3D surface reconstruction', *In Proc. SPIE*, vol. 3023, pp. 191-202.
- Hemayed, EE, Yamany, SM, Seales, WB & Farag, AA 1997, 'Three Dimensional Model Building In Computer Vision (II)', Technical document, University of Louisville.
- Henrich, D 1993, 'Space-efficient Region Filling in Raster Graphics', *The Visual Computer: An International Journal of Computer Graphics*, .

## Bibliography

- Hild, H, Haala, N & Fritsch, D 2000, 'A Strategy for Automatic Image to Map Registration', *International Archives of Photogrammetry and Remote Sensing*, vol. 33, no. B2; part 2, pp. 287-294.
- Hildreth, EC 1985, Edge Detection, *Massachusetts Institute of Technology*, Cambridge.
- Horchler, AD, Reeve, RE, Webb, BH & Quinn, RD 2003, 'Robot Phonotaxis in the Wild: a Biologically Inspired Approach to Outdoor', *11th International Conference on Advanced Robotics*, pp. 1749-1756.
- Horn, BKP 1988, 'Parallel networks for machine vision', Technical document, Massachusetts Institute of Technology.
- Horn, BKP & Schunck, BG 1981, 'Determining Optical Flow', *Artificial Intelligence*, vol. 16, no. 1-3, pp. 185-203.
- Hu, H & Gan, JQ 2005, 'Sensors and Data Fusion Algorithms in Mobile Robotics', Technical document, University of Essex.
- Huang, H, Maire, F & Keeratipranon, N 2005, '*Uncertainty Analysis of a Landmark Initialization Method for Simultaneous Localization and Mapping*', Proceedings of Australian Conference on Robotics and Automation.
- Huang, J, Kumar, SR, Mitra, M, Zhu, W-J & Zabih, R 1999, 'Spatial Color Indexing and Applications', *International Journal of Computer Vision*, vol. 35, no. 3, pp. 245-268.
- Huang, KS & Trivedi, MM 1998, 'Networked omnivision arrays for intelligent environment', *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation*, vol. 4479, pp. 129-134.
- Hubel, PM, Liu, J & Guttosch, RJ 2004, 'Spatial Frequency Response of Color Image Sensors: Bayer Color Filters and Foveon X3', *In Proc. SPIE*, vol. 5301, pp. 402-407.
- Huntsberger, T 2001, 'Biologically inspired autonomous rover control', *Autonomous Robots*, vol. 11; part 3, pp. 341-346.
- Hutchinson, S, Hager, G & Corke, P 1996, 'A tutorial on visual servo control', *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651-670.
- Huttenlocher, DP, Klanderman, GA & Rucklidge, WJ 1993, 'Compaing Images Using the Hausdorff Distance', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850-863.
- Iocchi, L & Konolige, K 1998, '*A Multiresolution Stereo Vision System for Mobile Robots*', In AIA Workshop.
- Irani, M & Anandan, P 1996, 'Parallax Geometry of Pairs of Points for 3D Scene Analysis', *Lecture Notes in Computer Science*, vol. 1064, pp. 17-30.
- Irani, M, Rousso, B & Peleg, S 1994, 'Recovery of Ego-Motion Using Image Stabilization', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 454-460.
- Irani, M, Rousso, B & Peleg, S 1997, 'Recovery of ego-motion using region alignment', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.

## Bibliography

- 19, no. 3, pp. 268-272.
- Isard, M & Blake, A 1998, 'CONDENSATION - Conditional Density Propagation for Visual Tracking', *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5-28.
- Ishiguro, H 1998, 'Development of low-cost compact omnidirectional vision sensors and their applications', *Proc. Int. Conf. Information systems, analysis and synthesis*, pp. 433-439.
- Ishiguro, H & Tsuji, S 1996, 'Image-Based Memory of Environment', *In Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 634-639.
- Jenkin, MRM & Jepson, A 1994, 'Detecting Floor Anomalies', *In Proc. British Machine Vision Conference*, pp. 731-740.
- Jensfelt, P 2000, 'Feature based condensation for mobile robot localization', *In IEEE Intl. Conf. on Robotics and Automation*, vol. 3, pp. 2531-2537.
- Jensfelt, P 2001, 'Approaches to Mobile Robot Localization in Indoor Environments', Thesis, Royal Institute of Technology.
- JR Kerr Automation Engineering 2005, 'PIC-SERVO Motor Control Board datasheet', Technical document, JR Kerr Automation Engineering.
- JR Kerr Automation Engineering 2005, 'PIC-Servo / PIC-Enc Servo Motion Control Chipset datasheet', Technical document, JR Kerr Automation Engineering.
- Kalman, RE 1960, 'A New Approach to Linear Filtering and Prediction Problem', *Journal of Basic Engineering*, vol. 82, series D, pp. 35-45.
- Kalviainen, H & Hirvonen, P 1995, 'Connective Randomized Hough Transform', *In Proc. 9th Scandinavian Conference on Image Analysis*, edit 9, vol. 2, pp. 1029-1036.
- Kang, SB & Szeliski, R 1997, '3-D scene data recovery using omnidirectional multibaseline stereo', *International Journal of Computer Vision*, vol. 25, no. 2, pp. 167-183.
- Katz, R, Melkumyan, N, Guivant, J, Bailey, T & Nebot, E 2005, '3D Sensing Framework for Outdoor Navigation', *In Proc. of the 2005 Australasian Conference on Robotics & Automation*.
- Kelly, A 1994, 'Concept Design of A Scanning Laser Rangefinder for Autonomous Vehicles', Technical document, CMU Robotics Institute.
- Kelly, SD, R & Murray, M 1994, 'Geometric Phases and Robotic Locomotion', Technical document, California Institute of Technology.
- Kim, J 2004, 'A framework for roadmap-based navigation and sector-based localization of mobile robots', Thesis, Texas A&M University.
- Kleeman, L 1992, 'Optimal Estimation of Position and Heading for Mobile Robots Using Ultrasonic Beacons and Dead-reckoning', *In Proc. IEEE International Conference on Robotics and Automation*, pp. 2582-2587.
- Kleeman, L 1999, 'Real time mobile robot sonar with interference rejection', *Sensor review*, vol. 19, no. 3, pp. 214-221.

## Bibliography

- Kleeman, L 2003, 'Advanced sonar sensing', *Springer Tracts in Advanced Robotics*, vol. 6, pp. 485-498.
- Kleeman, L & Kuc, R 1995, 'Mobile Robot Sonar for Target Localization and Classification', *International Journal of Robotics Research*, vol. 14, no. 4, pp. 295-318.
- Kleinberg, JM 1994, 'The localization problem for mobile robots', *Proceedings 35th Annual Symposium on Foundations of Computer Science*, vol. 35, pp. 521-531.
- Krootjohn, S 2007, 'Video image processing using MPEG Technology for a mobile robot', Thesis, Vanderbilt University.
- Krotkov, E 1991, 'Active Perception of Material and Shape by a Walking Robot', *Fifth International Conference on Advanced Robotics*, pp. 37-42.
- Kuipers, B 1978, 'Modeling spatial knowledge', *Cognitive Science*, vol. 2, no. 2, pp. 129-153.
- Kuipers, B & Byun, Y-T 1991, 'A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations', *Journal of Robotics and Autonomous Systems*, vol. 8, pp. 47-63.
- Kuipers, BJ & Levitt, TS 1988, 'Navigation and mapping in large-scale space', *AI Magazine*, vol. 9, pp. 25-43.
- Kumar, R, Anandan, P & Hanna, K 1994, 'Direct recovery of shape from multiple views: a parallax based approach', *International Conference on Pattern Recognition*, edit 12/1, pp. 685-685.
- Kwon, YD & Lee, JS 1995, 'An Obstacle Avoidance Algorithm For Mobile Robot: The Improved Weighted Safety Vector Field Method', *Proceedings of the IEEE 10th International Symposium on Intelligent Control*, pp. 441-446.
- Lacroix, S & Dudek, G 1997, 'On the identification of sonar features', *In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 592-599.
- Lamport, L 1978, 'Time, clocks, and the ordering of events in a distributed system', *Communications of the ACM*, vol. 21, no. 7, pp. 558-565.
- Lamport, L, Shostak, R & Pease, M 1982, 'The Byzantine generals problem', *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382-401.
- Latombe, J-C 1999, 'Motion planning: A journey of robots, molecules, digital actors, and other artifacts', *International Journal of Robotics Research*, vol. 18, no. 11, pp. 1119-1128.
- Lengyel, J, Reichert, M, Donald, BR & Greenberg, DP 1990, 'Real-time robot motion planning using rasterizing computer graphics hardware', *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pp. 327-335.
- Leonard, JJ & Durrant-Whyte, HF 1992, Directed sonar sensing for mobile robot navigation, *Kulwer Academic Publishers*, Massachusetts.
- Lewis, RA & Johnston, AR 1977, 'A Scanning Laser Rangefinder for a Robotic Vehicle', *In Proc. International Conference on Artificial Intelligence*, vol. 1, pp.

## Bibliography

762-768.

- Li, S & Hayashi, A 1998, 'Navigation by Integrating Iconic and GPS Information', *In Proc. IEEE International Conference on Intelligent Vehicles*, pp. 213-218.
- Liao, H-YM, Ko, M-T, Hsieh, J-W & Fan, K-C 1997, 'A new wavelet-based edge detector via constrained optimization', *Image and Vision Computing*, vol. 15, no. 7, pp. 511-527.
- Lindenberg, T 1996, 'Edge detection and ridge detection with automatic scale selection', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 465-470.
- Low, KH, Leow, WK & Ang, MH 2002, 'A hybrid mobile robot architecture with integrated planning and control', *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pp. 219-226.
- Lowe, DG 1987, 'Three-dimensional object recognition from single two-dimensional images', *Artificial Intelligence*, vol. 31, no. 3, pp. 355-395.
- Lowe, DG 1999, 'Object recognition from local scale-invariant features', *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150-1157.
- Lowe, DG 2004, 'Distinctive Image Features from Scale-Invariant Keypoints', *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110.
- Lu, F & Milios, E 1997, 'Robot pose estimation in unknown environments by matching 2D range scans', *Journal of Intelligent and Robotic Systems*, vol. 18, no. 3, pp. 249-275.
- Lucas, BD & Kanade, T 1981, 'An iterative image registration technique with an application to stereo vision', *International joint conference on artificial intelligence*, vol. 81, pp. 674-679.
- Mackenzie, P & Dudek, G 1994, 'Precise Positioning Using Model-Based Maps', *In Proc. IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1615-1621.
- MacLean, WJ, Jepson, AD & Frecker, RC 1994, 'Recovery of Egomotion and Segmentation of Independent Object Motion Using the EM Algorithm', *In Proceedings of the 5th British Machine Vision Conference*, pp. 175-184.
- Mandelbaum, R 1995, 'Sensor processing for mobile robot localization, exploration and navigation', Thesis, University of Pennsylvania.
- Marchand, E & Chaumette, F 2005, 'Features tracking for visual servoing purpose', *Robotics and Autonomous Systems*, vol. 52, no. 1, pp. 53-70.
- Mark, W, Fontijne, D, Dorst, L & Grown, FCA 2002, 'Vehicle egomotion estimation with geometric algebra', *In Proceedings IEEE Intelligent Vehicle Symposium*, vol. 1, pp. 18-20.
- Marsland, S, Nehmzow, U & Duckett, T 2001, 'Learning to select distinctive landmarks for mobile robot navigation', *Robotics and Autonomous Systems*, vol. 37, no. 4, pp. 241-260.
- Martin, MC & Moravec, HP 1996, 'Robot evidence grids', Technical document,

## Bibliography

Carnegie Mellon University.

- Martin, TL & Siewiorek, DP 1996, 'A Power Metric for Mobile Systems', *Proceedings of the 1996 international symposium on Low power electronics and design*, pp. 37-42.
- Masoud, SA & Masoud, AA 2000, 'Constrained Motion Control Using Vector Potential Fields', *IEEE Transaction on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 30, no. 3, pp. 251-272.
- Mata, M, Armingol, JM, De La Escalera, A & Salichs, MA 2002, 'Learning visual landmarks for mobile robot navigation', *In Proceedings of the 15th World congress of the International Federation of Automatic Control*, vol. 21, pp. 26-31.
- Matas, J, Burianek, J & Kittler, J 2000, 'Object Recognition using the Invariant Pixel-Set Signature', *In Proc. British Machine Vision Conference*, pp. 606-615.
- Mattavelli, M, Noel, V, & Amaldi, E 1999, 'A new approach for fast line detection based on combinatorial optimization', *In Proc. International Conference on Image Analysis and Processing*, pp. 168-173.
- Mattern, F 1989, 'Virtual time and global states of distributed systems', *Parallel and Distributed Algorithms*, pp. 215-226.
- Mattern, F, Mehl, H, Schoone, AA & Tel, G 1991, 'Global Virtual Time Approximation with Distributed Termination Detection Algorithms', Technical document, Utrecht University.
- Matthies, L, Szeliski, R & Kanade, T 1988, 'Kalman Filter-based Algorithms for Estimating Depth from Image Sequences', Technical document, Carnegie Mellon University.
- Mayer, HA, Schmidbauer, J & Stieglbauer, G 2001, 'EMMA - Architecture and Subsystems of a Mobile Autonomous Robot with a Preference for Soccer', *In Proc. WSES International Conference on Robotics, Distance Learning and Intellignt Communication Systems*, pp. 310-316.
- McCarthy, C & Barnes, N 2003, 'Performance of temporal filters for optical flow estimation in mobile robot corridor centring and visual odometry', *Proceedings of the 2003 Australasian Conference on Robotics and Automation*.
- McCarthy, CD 2005, 'Performance of Optical Flow Techniques for Mobile Robot Navigation', Thesis, University of Melbourne.
- McLurkin, JD 2004, 'Stupid Robot Tricks: A Behavior-Based Distributed Algorithm Library for Programming Swarms of Robots', Thesis, Massachusetts Institute of Technology.
- Menegatti, E, Maeda, T & Ishiguro, H 2004, 'Image-based memory for robot navigation using properties of omnidirectional images', *Robotics and Autonomous Systems*, vol. 47, no. 4, pp. 251-267.
- Merke, A, Welker, S & Riedmiller, M 2004, 'Line Based Robot Localization under Natural Light Conditions', *In ECAI 2004 Workshop on Agents in Dynamic and Real Time Environments*.
- Miura, J, Uozumi, H & Shirai, Y 1999, 'Mobile Robot Motion Planning Considering



## Bibliography

- the Motion Uncertainty of Moving Obstacles', *In Proc. IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, pp. IV-692-IV-697.
- Mondada, F & Franzi, E 1993, 'Biologically inspired mobile robot control algorithms', *In NFP-PNR 23 Symposium*, pp. 47-60.
- Montemerlo, M 2002, 'FastSLAM: A factored solution to the simultaneous localization and mapping problem', *In Proceedings of the AAAI National Conference on Artificial Intelligence*, no. 18, pp. 593-598.
- Murray, D & Jennings, C 1997, 'Stereo vision based mapping and navigation for mobile robots', *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1694-1699.
- Murray, D & Little, J 2000, 'Using real-time stereo vision for mobile robot navigation', *Autonomous Robots*, vol. 8, no. 2, pp. 161-171.
- Nagatani, K, Choset, H & Thrun, S 1998, 'Towards Exact Localization without Explicit Localization with the Generalized Voronoi Graph', *In Proc. IEEE International Conference on Robotics and Automation*, no. 1, pp. 342-348.
- Nagchaudhuri, A 2002, 'Robotics and Machine Vision for Introduction to Flexible Automation to Engineering Undergraduates', *In Proc. 32nd ASEE/IEEE Frontiers in Education Conference*, pp. 23-28.
- Narkhede, S & Golshani, F 2004, 'Stereoscopic imaging: a real-time, in depth look', *IEEE Potentials*, vol. 23; part 1, pp. 38-42.
- Navalpakkam, V & Itti, L 2005, 'Modeling the influence of task on attention', *Vision Research*, vol. 45, no. 2, pp. 205-231.
- Nayar, SK 1997, 'Omnidirectional video camera', *Proc. DARPA Image Understanding Workshop*, vol. 1, pp. 235-242.
- Nayar, SK, Murase, H & Nene, SA 1994, 'Learning, Positioning, and Tracking Visual Appearance', *In Proc. IEEE International Conference on Robotics and Automation*, pp. 3237-3244.
- Negenborn, R 2003, 'Robot Localization and Kalman Filters On finding your position in a noisy world', Thesis, Utrecht University.
- Nehmzow, U, Gelder, D & Duckett, T 2000, 'Automatic selection of landmarks for mobile robot navigation', Technical document, UMCSUniversity of Manchester.
- Neira, J, Tardos, JD, Horn, H & Schmidt, G 1999, 'Fusing range and intensity images for mobile robot localization', *IEEE Transactions on Robotics and Automation*, vol. 15, no. 1, pp. 76-84.
- Ng, TW 2003, 'The optical mouse as a two-dimensional displacement sensor', *Sensors and Actuators*, vol. 107, no. 1, pp. 21-25.
- Ng, TW & Carne, M 2007, 'Optical mouse digital speckle correlation', *Optics Communications*, vol. 280, no. 2, pp. 435-437.
- Novak, CL & Shafer, SA 1992, 'Anatomy of a Color Histogram', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 599-605.
- Oh, S, Zelinsky, A & Taylor, K 2000, 'Autonomous battery recharging for indoor

## Bibliography

- mobile robots*', Proceedings of Australian Conference on Robotics and Automation.
- Olson, CF 2000, 'Probabilistic self-localization for mobile robots', *IEEE Transactions on Robotics and Automation*, vol. 16, no. 1, pp. 55-66.
- Oren, M & Nayar, SK 1995, 'Generalization of the Lambertian Model and Implications for Machine Vision', *International Journal of Computer Vision*, vol. 14, no. 3, pp. 227-251.
- Ostrowski, Jim 1999, 'Steering for a class of dynamic nonholonomic systems', *IEEE Transactions on Automatic Control*, vol. 45; part 8, pp. 1492-1497.
- Ostrowski, JP, Desai, JP & Kumar, V 1997, 'Optimal Gait Selection for Nonholonomic Locomotion Systems', *IEEE International Conference on Robotics and Automation*, no. 1, pp. 786-791.
- Paletta, L, Fritz, G & Selfert, C 2005, 'Q-Learning of Sequential Attention for Visual Object Recognition from Informative Local Descriptors', *In Proc. 22nd International Conference on Machine Learning*, vol. 22, pp. 649-656.
- Paromtchik, IE & Nassal, UM 1995, 'Reactive Motion Control for an Omnidirectional Mobile Robot', *Proc. of the Third European Control Conference*, pp. 5-8.
- Pass, G, Zabih, R & Miller, J 1996, 'Comparing images using color coherence vectors', *Proceedings of the fourth ACM international conference on Multimedia*, pp. 65-73.
- Peters, RA II 1995, 'A New Algorithm for Image Noise Reduction using Mathematical Morphology', *IEEE Transactions on Image Processing*, vol. 4, no. 3, pp. 554-568.
- Peters, RA II & Strickland, RN 1990, 'Image complexity metrics for automatic target recognizers', *Automatic Target Recognizer System and Technology Conference*, pp. 1-17.
- Peters, RA II, Bishay, M & Rogers, T 1996, 'On the Computation of the Log-Polar Transform', Technical document, Vanderbilt University.
- Pfister, ST, Kriechbaum, KL, Roumeliotis, SI & Burdick, JW 2002, 'Weighted Range Sensor Matching Algorithms for Mobile Robot Displacement Estimation', *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1667-1674.
- Phong, BT 1975, 'Illumination for Computer Generated Pictures', *Communications of the ACM*, vol. 18, no. 6, pp. 311-317.
- Pitteway, MLV & Watkinson, DJ 1980, 'Bresenham's algorithm with Grey scale', *Communications of the ACM*, vol. 23, no. 11, pp. 625-626.
- Polaroid 1995, '600 Series Instrument Grade Electrostatic Transducer', Technical document, Polaroid.
- Poulin, P & Fournier, A 1990, 'A model for anisotropic reflection', *ACM SIGGRAPH Computer Graphics*, vol. 24, no. 4, pp. 273-282.
- Pratt, J, Dilworth, P & Pratt, G 1997, 'Virtual model control of a bipedal walking robot', *IEEE Conference on Robotics and Automation*, no. 1, pp. 193-198.

## Bibliography

- Quan, L 1996, 'Self-calibration of an affine camera from multiple views', *International Journal of Computer Vision*, vol. 19, no. 1, pp. 93-105.
- Quan, L & Kanade, T 1997, 'Affine structure from line correspondences with uncalibrated affine cameras', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 8, pp. 834-845.
- Quinn, RD, Nelson, GM, Bachmann, RJ, Kingsley, DA, Offi, J & Ritzmann, RE 2001, 'Insect designs for improved robot mobility', *Proceedings of 4 th Int. Conf. On Climbing and Walking Robots*, pp. 69-76.
- Rajendran, S & Huber, M 2004, 'Developing Task Specific Sensing Strategies Using Reinforcement Learning', *In Proc. 17th International FLAIRS Conference*, pp. 738-743.
- Redlick, FP, Jenkin, M & Harris, LR 2001, 'Humans can use optic flow to estimate distance of travel', *Vision Research*, vol. 41, no. 2, pp. 213-219.
- Rekleitis, LM, Dudek, G & Milios, EE 1997, 'Multi-robot exploration of an unknown environment, efficiently reducing the odometry error', *In Proc. of the International Joint Conference on Artificial Intelligence*, vol. 15, no. 2, pp. 1340-1345.
- Remolina, E & Kuipers, B 2004, 'Towards a general theory of topological maps', *Artificial Intelligence*, vol. 152, no. 1, pp. 47-104.
- Ritter, GX & Wilson, JN 1996, *Handbook of Computer Vision Algorithms in Image Algebra*, CRC press, United States.
- Rives, P, Pissard-Gibollet, R & Kapellos, K 1993, 'Development of a reactive mobile robot using real time vision', *3rd International Symposium on Experimental Robotics*, pp. 486-500.
- Rosselot, D & Hall, EL 2004, 'Processing real-time stereo video for an autonomous robot using disparity maps and sensor fusion', *In Proc. SPIE*, vol. 5608, pp. 70-78.
- Rosten, E & Drummond, T 2005, 'Fusing Points and Lines for High Performance Tracking', *Tenth IEEE International Conference on Computer Vision*, vol. 2, pp. 1508-1515.
- Roumeliotis, SI & Bekey, GA 1997, 'An extended Kalman filter for frequent local and infrequent global sensor data fusion', *in SPIE International Symposium on Intelligent Systems and Advanced Manufacturing*, pp. 11-22.
- Roy, N & Dudek, G 2001, 'Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations', *Autonomous Robots*, vol. 11; part 2, pp. 117-136.
- Schauer, C & Gross, H-M 2001, 'A Model of Horizontal 360 Object Localization based on Binaural Hearing and Monocular Vision', *Lecture notes in computer science*, pp. 1141-1146.
- Schiele, B & Crowley, JL 1994, 'A comparison of position estimation techniques using occupancy grids', *In Proc. of the IEEE International Conference on Robotics & Automation*, vol. 12, no. 3/4, pp. 1628-1634.
- Schilling, A 1991, 'A New Simple and Efficient Antialiasing with Subpixel Masks',

## Bibliography

- Computer Graphics*, vol. 25, no. 4, pp. 133-141.
- Schwarz, R & Mattern, F 1994, 'Detecting Causal Relationships in Distributed Computations: in Search of the Holy Grail', *Distributed computing*, vol. 7, no. 3, pp. 149-174.
- Se, S, Lowe, D & Little, J 2002, 'Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks', *International Journal of Robotics Research*, vol. 21; part 8, pp. 735-758.
- Se, S, Lowe, D & Little, J 2002, 'Global Localization using Distinctive Visual Features', *In Proc. International Conference on Intelligent Robots and Systems*, pp. 226-231.
- Se, S, Lowe, D & Little, J 2002, 'Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks', *International Journal of Robotics Research*, vol. 21; part 8, pp. 735-758.
- Seitz, SM & Dyer, CR 1997, 'Photorealistic scene reconstruction by voxel coloring', *Proc. CVPR*, pp. 1067-1073.
- SensComp. Inc. 2004, '6500 Series Ranging Modules', Technical document, SensComp Inc..
- Shaked, D, Yaron, O & Kiryati, N 1994, 'Deriving Stopping Rules for the Probabilistic Hough Transform by Sequential Analysis', *In Proc. 12th IAPR International Conference on Pattern Recognition*, vol. 2, pp. 229-234.
- Shakernia, O, Vidal, R & Sastry, S 2003, 'Omnidirectional egomotion estimation from back-projection flow', *In Proc. Conference on Computer Vision and Pattern Recognition Workshop*, vol. 7, pp. 82-87.
- Shannon, CE 1998, 'Communication in the Presence of Noise', *Proceedings of the IEEE*, vol. 86, no. 2, pp. 447-457.
- Sharon, E, Brandt, A & Basri, R 2001, 'Segmentation and boundary detection using multiscale intensity measurements', *In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I-469-I476.
- Shatkay, H & Kaelbling, LP 1997, 'Learning Topological Maps with Weak Local Odometric Information', *In Proceedings of IJCAI-97*, vol. 15, no. 2, pp. 920-929.
- Shen, J & Hu, H 2004, 'Mobile robot navigation through digital landmarks', *In Proc. of the 10th Chinese Automation and Computing Conf*, pp. 151-156.
- Shen, WM, Adibi, J, Adobbati, R, Cho, B, Erdem, A, Moradi, H, Salemi, B, & Tejada, S 1998, 'Building Integrated Mobile Robots for Soccer Competition', *Proceedings 1998 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2613-2618.
- Shi, J & Malik, J 2000, 'Normalised Cuts and Image Segmentation', *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888-905.
- Shi, J & Tomasi, C 1994, 'Good features to track', *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593-600.
- Sibley, GT, Rahimi, MH, Sukhatme, GS 2002, 'Robomote: A Tiny Mobile Robot

## Bibliography

- Platform for Large-Scale Ad-hoc Sensor Networks', *In Proc. IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1143-1148.
- Sim, R 1998, 'Mobile Robot Localization Using Learned Landmarks', Thesis, McGill University.
- Sim, R & Dudek, G 1998, 'Mobile robot localization from learned landmarks', *In Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems*, vol. 2, pp. 1060-1065.
- Simmons, R & Koenig, S 1995, 'Probabilistic robot navigation in partially observable environments', *In Proceedings of International Joint Conference on Artificial Intelligence*, vol. 14, no. 2, pp. 1080-1087.
- Simoncelli, EP 1996, 'Noise removal via Bayesian wavelet coring', *In Proc. 3rd IEEE International Conference Image Processing*, vol. 1, pp. 379-382.
- Smith, P, Reid, I & Davison, A 2006, 'Real-Time Monocular SLAM with Straight Lines', *Proc 17th British Machine Vision Conference*, pp. 17-26.
- Smith, SM & Brady, JM 1997, 'Susan - a new approach to low level image processing', *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45-78.
- Soyer, C, Bozman, I & Istefanopulos, Y 2003, 'Attentional Sequence-Based Recognition: Markovian and Evidential Reasoning', *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 33, no. 6, pp. 937-950.
- Spacek, L 2005, 'A Catadioptric Sensor with Multiple Viewpoints', *Robotics and Autonomous Systems*, vol. 51, no. 1, pp. 3-15.
- Spacek, L & Burbridge, C 2007, 'Instantaneous Robot Self-Localisation and Motion Estimation with Omnidirectional Vision', *Robotics and Autonomous Systems*, vol. 55, no. 9, pp. 667-674.
- Stachniss, C & Burgard, W 2005, 'Mobile Robot Mapping and Localization in Non-Static Environments', *In Proc. The National Conference on Artificial Intelligence*, vol. 20, no. 3, pp. 1324-1329.
- Stocker, AA 2002, 'An improved 2-D optical flow sensor for motion segmentation', *IEEE International Symposium on Circuits and Systems*, part 2, pp. II-332-335.
- Stricker, M & Swain, M 1994, 'The capacity of color histogram indexing', *In Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 704-708.
- Sundareswaran, V, Bouthemy, P & Chaumette, F 1994, 'Visual servoing using dynamic image parameters', Technical document, Institut de Recherche en Informatique et Systemes Aleatoires.
- Taylor, CJ & Kriegman, DJ 1998, 'Vision-Based Motion Planning and Exploration Algorithms for Mobile Robots', *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 417-426.
- Taylor, T, Geva, S & Boles, WW 2006, 'Using Camera Tilt to Assist with Localisation', *In Proc. 3rd International Conference on Autonomous Robots and Agents*.
- Thau, RS 1997, 'Reliably Mapping a Robot's Environment using Fast Vision and

## Bibliography

- Local, but not Global, Metric Data', Thesis, Massachusetts Institute of Technology.
- Thompson, WB, & Pick Jr, HL 1993, 'Vision-Based Navigation', *Image Understanding Workshop: Proceedings of a Workshop Held in Washington*, pp. 127-134.
- Thompson, WB, Henderson, TC, Colvin, TL, Dick, LB & Valiquette, CM 1993, 'Vision-based localization', *DARPA Image Understanding Workshop*, pp. 491-498.
- Thorpe, CF 1984, 'Path Relaxation: Path Planning for a Mobile Robot', Technical document, Carnegie-Mellon University.
- Thrun, S 1998, 'Learning Maps for Indoor Mobile Robot Navigation', *Artificial Intelligence*, vol. 99, no. 1, pp. 21-71.
- Thrun, S 2000, 'Probabilistic algorithms in robotics', *AI Magazine*, vol. 21; part 4, pp. 93-110.
- Thrun, S 2001, 'A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots', *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 335-377.
- Thrun, S 2002, 'Robotic Mapping: A Survey', Technical document, Carnegie Mellon University.
- Thrun, S & Bucken, A 1996, 'Integrating Grid-Based and Topological Maps for Mobile Robot Navigation', *In Proc. 13th National Conference on Artificial Intelligence*, pp. 944-950.
- Thrun, S, Burgard, W & Fox, D 1998, 'A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots', *Autonomous Robots*, vol. 5, no. 3/4, pp. 253-271.
- Thrun, S, Fox, D, Burgard, W & Dellaert, F 2001, 'Robust Monte Carlo Localization for Mobile Robots', *Artificial Intelligence*, vol. 128, no. 1, pp. 99-141.
- Torre, V & Poggio, T 1986, 'On edge detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 147-163.
- Tsai, RY 1987, 'A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses', *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, pp. 323-344.
- Tucakov, V, Sahota, M, Murray, D, Mackworth, A, Little J, Kingdon, S, Jennings, C & Barman, R 1997, 'Spinoza: A stereoscopic visually guided mobile robot', *In Proc. of Hawaii International Conference on Systems Sciences*, vol. 30, no. 5, pp. 188-197.
- Turkowski, K 1990, 'Filters for Common Resampling Tasks', Technical document, Apple Computer.
- Van Aken, J & Novak, M 1985, 'Curve-Drawing Algorithms for Raster Displays', *ACM Transactions on Graphics*, vol. 4, no. 2, pp. 147-169.
- Varveropoulos, V 2000, 'Robot Localization and Map Construction Using Sonar Data', *The Rossum Project*, <http://rosum.sourceforge.net>.

## Bibliography

- Vassallo, RF, Santos-Victor, J & Schneebeli, HJ 2002, 'A general approach for egomotion estimation with omnidirectional images', *In Proc. IEEE Workshop on Omnidirectional Vision*, pp. 97-103.
- Vilmanis, Y 2005, 'Intelligent Robot System', Thesis, Flinders University.
- Voo, CY 2000, 'Low Level Driving Routines for the OMNI-Directional Robot', Thesis, University of Western Australia.
- Wallner, F, Schiele, B & Crowley, JL 1997, 'Position Estimation for a Mobile Robot from Principal Components of Laser Data', *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 215-224.
- Wang, H & Brady, M 1995, 'Real-time corner detection algorithm for motion estimation', *Image and Vision Computing*, vol. 13, pp. 695-703.
- Ward, K & Zelinsky, A 1997, 'An Exploratory Robot Controller which Adapts to Unknown Environments and Damaged Sensors', *In Proc. International Conference on Field and Service Robots*, pp. 477-484.
- Wei, G, Wetzler, C & von Puttkamer, E 1994, 'Keeping Track of Position and Orientation of Moving Indoor Systems by Correlation of Ranger-Finder Scans', *In Proc. IEEE Conf*, pp. 595-601.
- Welch, G & Bishop, G 1995, 'An Introduction to the Kalman Filter', Technical document, University of North Carolina at Chapel Hill.
- Werman, M, Banerjee, S, Roy, SD & Qiu, M 1999, 'Robot Localization using Uncalibrated Camera Invariants', *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 353-359.
- Whaite, P & Ferrie, PF 1997, 'Autonomous exploration: Driven by uncertainty', , vol. 19, no. 3, pp. 193-205.
- Whelan, PF & Molloy, D 2001, *Machine Vision Algorithms in Java*, Springer-Verlag, London.
- Wijk, O 2001, 'Triangulation Based Fusion of Sonar Data with Application in Mobile Robot Mapping and Localization', Thesis, Royal Institute of Technology.
- Wijk, O & Christensen, HI 2000, 'Triangulation-Based Fusion of Sonar Data with Application in Robot Pose Tracking', *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 740-752.
- Wijk, O, Jensfelt, P & Wahlberg, B 1997, 'Sensor Fusion for Mobile Robot Navigation - A First Subjective Discussion', Technical document, Royal Institute of Technology Stockholm.
- Williams, SB 2001, 'Efficient Solutions to Autonomous Mapping and Navigation Problems', Thesis, University of Sydney.
- Winters, N & Santos-Victor, J 1999, 'Mobile robot navigation using omni-directional vision', *In Proc. 3rd Irish Machine Vision and Image Processing Conf*, pp. 151-166.
- Winters, N, Gaspar, J, Lacey, G & Santos-Victor, J 2000, 'Omni-directional vision for robot navigation', *In Proc. IEEE Workshop on Omnidirectional Vision*, pp. 21-28.

## Bibliography

- Wolf, J, Burgard, W & Burkhardt, H 2002, 'Robust Vision-Based Localization for Mobile Robots using an Image Retrieval System Based on Invariant Features', *In Proc. of the IEEE International Conference on Robotics & Automation*, vol. 1, pp. 359-365.
- Wu, X 1991, 'An Efficient Antialiasing Technique', *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, vol. 25, no. 4, pp. 143-152.
- Xu, W, Jenkin, M & Lesperance, Y 2006, 'A Multi-Channel Algorithm for Edge Detection Under Varying Lighting Conditions', *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 1885-1892.
- Yagi, Y 1999, 'Omnidirectional Sensing and Its Applications', *IEICE Transaction on Information and Systems E series D*, vol. 82, no. 3, pp. 568-579.
- Yamauchi, B, Schultz, A & Adams, W 1998, 'Mobile robot exploration and map-building with continuous localization', *IEEE/RSJ International Conference on Robotics and Automation*, no. 4, pp. 3175-3720.
- Yang, S-N & Lee, R-R 1994, 'Parallel Quadtree Construction and Manipulation Algorithms on Hypercubes', *In Proc. International Conference on Computing and Information*, vol. 94, pp. 847-866.
- Zelek, JS 1995, 'Dynamic path planning', *Proceedings of IEEE Conference on Systems, Man and Cybernetics*, vol. 2, pp. 1285-1290.
- Zelinsky, A & Yuta, S 1993, 'A unified approach to planning, sensing and navigation for mobile robots', *Third International Symposium on Experimental Robotics*, pp. 444-455.
- Zhang, Z & Kodagoda, KR 2005, 'A Monocular Vision Based Localizer', .
- Zhang, Z, Deriche, R, Faugeras, O & Luong, QT 1994, 'A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry', Technical document, Unite de recherche INRIA Sophia-Antipolis.
- Zhao, L 1998, 'Recursive Estimation of 3-D Motion Trajectories from Image Sequences Using Measurements of Feature Point Positions and Optical Flow', Thesis, Queen's University.
- Zimmer, UR 1995, 'Self-localization in dynamic environments', IEEE/SOFT International Workshop.
- Zimmer, UR 1996, 'Robust world-modeling and navigation in a real world', *Neurocomputing*, vol. 13, pp. 2-4.
- Ziou, D & Tabbone, S 1997, 'Edge detection techniques - an overview', *International Journal of Pattern Recognition and Image Analysis*, vol. 8, no. 4, pp. 537-559.
- Zitnick, CL & Kanade, T 2000, 'A Cooperative Algorithm for Stereo Matching and Occlusion Detection', *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22; part 7, pp. 675-684.
- Zlatev, J & Balkenius, C 2001, 'Introduction: Why "Epigenetic Robotics"?', *Proceedings of the first international workshop on epigenetic robotics: Modeling*



## Bibliography

*cognitive development in robotic systems*, vol. 85, pp. 1-4.