



FLINDERS UNIVERSITY

FINAL THESIS

**Investigating Robotic Designs to address the
ever-increasing space debris**

Sai Juturu

ID: 2182059

FAN: jutu0001

Topic Code: ENGR9700 (A-D) - Masters Thesis (18 units)

Degree: Bachelor of Engineering (Robotics) (Honours) / Master of Engineering
(Electronic)

Supervised by

Principal/Primary Supervisor Mr. Pepe VELASQUEZ

Secondary Supervisor Dr. Sherry RANDHAWA

2 December 2022

1 Abstract

Since the dawn of space exploration in 1957, numerous structures such satellites and rockets, have been situated in Earth's orbit and space. Due to a lack of a system that deals with the disposal of now defunct structures (or their collision-resultant fragments), the amount of debris in space is estimated to have reached the Kessler benchmark. Thus, there is a need to conceptualize systems to improve the safety of all personnel and property in space. The thesis aims to evaluate the viability of image processing and machine learning techniques in the field of debris identification. The kernel-based program (image processing) was capable of detecting space debris, however the quality of the output was degraded due to preset baseline assumptions. Other factors that determined this program's inability to appropriately function in space was its dependency on color/pixel values; variables that cannot be replicated in actual space, more so since this code is tested upon clips from the movie "Gravity". The project then progresses to validate the applicability of machine learning in similar scenarios. Since the model's training and testing phases were comprised of only 200 images (along with being synthetic), the precision of 0.6 and recall of 0.67 indicate that the model developed for this thesis is not realistically feasible. However, the research did indicate that machine learning with better resources could be a potential solution working towards space debris detection. Subsequently, the thesis also dives into the field of robotic design, 3D modelling a hypothetical robot, named the WOMBAT, that would be theoretically capable of capturing/collecting space debris through the equipped net/sheet. Finally, the discussion gains a marketing perspective on introducing such a product, and the resistance it could receive due to its potential ability to gain access to inter-country security and surveillance data.

2 Declaration

I certify that this thesis:

1. Does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university.
2. And the research within will not be submitted for any other future degree or diploma without the permission of Flinders University.
3. And to the best of my knowledge and belief, does not contain any material previously published or written by another person except where due reference is made in the text.

.....

Signature of Student

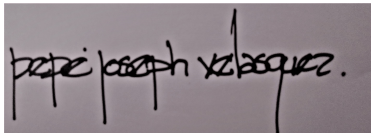
.....

Print Name of Student

.....

Date

I certify that I have read this thesis. In my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Bachelor of Engineering (Robotics) (Honours) / Master of Engineering (Electronic). Furthermore, I confirm that I have provided feedback on this thesis and the student has implemented it fully.



Signature of Principal Supervisor

.....

Print Name of Principal Supervisor

.....

Date

3 Acknowledgments

I would like to acknowledge the assistance provided to me throughout the duration of this thesis. Firstly, I express my gratitude to my primary supervisor, Mr. Pepe Velasquez, for all his help towards designing, exploring different research avenues, and making progress at various phases of the project. I would also like to appreciate Dr. Sherry Randhawa for taking up the secondary supervisor role behind this thesis. Lastly, I would like to thank Flinders University for providing me with this research opportunity.

Contents

| | | |
|----------|---|-----------|
| 1 | Abstract | 1 |
| 2 | Declaration | 2 |
| 3 | Acknowledgments | 3 |
| 4 | Introduction | 7 |
| 4.1 | Thesis Premise | 7 |
| 4.2 | Problem Statement: Background Information | 7 |
| 4.2.1 | GAP: Current Methods of Space Debris Disposal & Con- tainment | 9 |
| 4.3 | Thesis Project Scope | 10 |
| 5 | Literature Review | 12 |
| 5.1 | Investigating Potential Robotic Designs & Structures | 12 |
| 5.2 | Investigating Potential Means of Space Debris Identification | 15 |
| 5.3 | Section Overview | 17 |
| 6 | Methodology | 18 |
| 6.1 | Equipment Configurations | 18 |
| 6.1.1 | Hardware | 18 |
| 6.1.2 | Software | 18 |
| 6.2 | Primary Goal: Part 1 - Kernel-based Image Processing | 19 |
| 6.2.1 | Listing 1: Initializations, Capturing the video and Setting up the kernels | 19 |
| 6.2.2 | Listing 2: Resizing, Color conversion, Mask generation and manipulation, Contouring, and Displaying final output | 20 |
| 6.3 | Concluding Comments | 21 |
| 6.4 | Primary Goal: Part 2 - Machine Learning Algorithm | 21 |
| 6.4.1 | Virtual Environment | 21 |
| 6.4.2 | Listing 3: Initializations and Image Collection | 22 |
| 6.4.3 | Listing 4: Image Labelling | 23 |
| 6.4.4 | Listing 5: Part 1 - Setting Up the Training and Detection Algorithm | 24 |
| 6.4.5 | Listing 6: Part 2 - Setting Up the Training and Detection Algorithm (Continued) | 25 |
| 6.4.6 | Listing 7: Part 3 - Setting Up the Training and Detection Algorithm (Continued) | 25 |
| 6.4.7 | Listing 8: Part 4 - Setting Up the Training and Detection Algorithm (Continued) | 26 |

| | | |
|-----------|---|-----------|
| 6.4.8 | Listing 9: Part 5 - Running the Training Phase, Evaluation Phase and Setting up Checkpoints | 27 |
| 6.4.9 | Listing 10: Part 6 - Running the Testing Phase | 27 |
| 6.4.10 | Concluding Comments | 28 |
| 6.5 | Secondary Goal: 3D Modelling | 28 |
| 6.5.1 | WOMBAT: Design Phase | 28 |
| 6.5.2 | WOMBAT: Engineering Phase | 31 |
| 6.6 | Section Overview | 32 |
| 7 | Results | 33 |
| 7.1 | Kernel-based Image Processing | 33 |
| 7.2 | Machine Learning Algorithm | 34 |
| 7.3 | 3D Modelling & Animation | 36 |
| 7.4 | Section Overview | 37 |
| 8 | Discussion | 38 |
| 8.1 | Kernel-based Image Processing Program | 38 |
| 8.2 | Machine Learning Algorithm | 39 |
| 8.2.1 | Recorded Loss Curves | 39 |
| 8.2.2 | Recorded Precision - Mean Average Precision (mAP) | 41 |
| 8.2.3 | Recorded Recall | 41 |
| 8.2.4 | Comments | 42 |
| 8.3 | 3D Modelling & Animation | 43 |
| 8.4 | Market & Political Disputes | 43 |
| 8.5 | Section Overview | 44 |
| 9 | Conclusions | 45 |
| 10 | Future Work | 46 |
| 11 | Bibliography | 47 |
| 12 | Appendices | 50 |
| 12.1 | Appendix A: Graveyard Orbit | 50 |
| 12.2 | Appendix B: Earth's Orbits and Satellites | 50 |
| 12.3 | Appendix C: Code Listings | 51 |

List of Figures

| | | |
|----|--|----|
| 1 | Earth-based Test: Hyper-Velocity Debris Damage | 8 |
| 2 | The Space Cemetery: Point Nemo | 9 |
| 3 | Tzutalin’s Labelling Software/Tool | 23 |
| 4 | Kernel-based Image Processing Outputs | 33 |
| | (a) Scene 1 - Captured Action Frame | 33 |
| | (b) Scene 2 - Captured Action Frame | 33 |
| 5 | Machine Learning vs Astronaut Detection + Confidence Metrics . . | 34 |
| | (a) Astronaut Recognition - Image 1 | 34 |
| | (b) Astronaut Recognition - Image 2 | 34 |
| 6 | Machine Learning vs [Extras, Satellite] + Confidence Metrics | 34 |
| | (a) Extras Recognition Example | 34 |
| | (b) Satellite Recognition Example | 34 |
| 7 | Identifying two different objects: Example 1 - Astronaut + Satellite | 35 |
| 8 | Identifying two different objects: Example 2 - Astronaut + Debris . | 35 |
| 9 | WOMBAT: Schematic Model | 36 |
| 10 | WOMBAT: Full + Cross-sectional View | 36 |
| | (a) WOMBAT: Full 3D Model | 36 |
| | (b) WOMBAT: Cross-sectional View | 36 |
| 11 | Animation Frame: WOMBAT in Action | 37 |
| 12 | The 4 types of Loss Curves | 40 |
| | (a) Classification Loss Curve | 40 |
| | (b) Localization Loss Curve | 40 |
| | (c) Regularization Loss Curve | 40 |
| | (d) Total Loss Curve | 40 |
| 13 | Precision Curve over 6000 steps (mAP) | 41 |
| 14 | Recall Curve | 42 |
| 15 | The Graveyard Orbit | 50 |
| 16 | Earth’s Orbits and Satellites | 50 |

4 Introduction

4.1 Thesis Premise

The premise of this thesis revolves around approaching the Kessler phenomenon and the need to address the present space debris crisis. The 'Kessler' syndrome, proposed by the ex-NASA scientist and debris expert Donald J. Kessler, is a theory that states the exponentially increasing space debris will reach a peak value, an amount that will enable the existing quantity to continuously increase even without further man-made launches [Matignon & de Gouyon (2019)]. When defined in 1978, Don Kessler estimated approximately 30 to 40 years for the environment in space to achieve the Kessler syndrome and based from current NASA experts "we are already at critical mass in the low-Earth orbit"[Ratner (2018)].

4.2 Problem Statement: Background Information

In order to better substantiate the ever-increasing danger, it is important to keep track of the amount of debris introduced over the decades since 1957. Since the dawn of the space age, ranging from defunct satellites to rocket remains, it is recorded that there are over 22000 large objects orbiting the planet. The scenario only deteriorates when taking another 1 million random smaller objects, like discarded astronaut gear etc., into the total count. To better highlight the threat, a space environmental statistical report in 2021 compiling data since the beginning of space exploration provided an assessment on the current of debris within Earth's orbit [ESA (2021)]:

- # of recorded rocket launches since the start of space exploration in 1957: **6170** (does not account for failures)
- # of recorded successful satellites propagated/placed in orbit: **12470**
- # of previously stated satellites still suspended in orbit: **7840**
- # of satellites within the above subset that still function: **5100**
- regular # of objects/debris catalogued by Space Surveillance Networks: **30040**

- approximate # of events (such as collisions) that led to further fragmentation since 1957: **>630**
- accumulate mass of space debris/fragments/objects currently in orbit: **>9800 tonnes**
- a statistical estimate on debris quantity with respect to their size:
 - # of objects **> 10 cm = 36500**
 - # of **1cm < objects < 10 cm = 1 million**
 - # of **1mm < objects < 1 cm = 130 million**

The above data represents the current state of space-debris across the 14 different orbits enveloping planet Earth. The scope of this thesis shall revolve around the previously iterated "critical mass" in the Low Earth Orbit (LEO). LEO is among the most highly prioritized orbit as it is the most readily accessible for satellite placement, houses a large population of working astronauts and requires the lowest amount of energy to gain access to (as portrayed in Figure 16). LEO is also home to the International Space Station (ISS) and their crews who are in charge of maintaining and servicing subsequent satellites and space stations [UWA (2022)]. Figure 1 depicts how a relatively small fragment/debris is capable of substantial damage when travelling at hyper-velocity [ESA (2020)].

Figure removed due to copyright restriction

Figure 1: Earth-based Test: Hyper-Velocity Debris Damage

4.2.1 GAP: Current Methods of Space Debris Disposal & Containment

Having grounded the current situation, it is vital to descry the GAP as it establishes the purpose behind this thesis project, that being the containment and disposal of space debris. The current methods being implemented to 'dispose' of space junk are not relatively effective, one of the two methods only sustaining/speeding up the rate at which the quantity of space debris achieves the Kessler syndrome. The two means being as follows: (1) **Point Nemo** (also labelled as the 'space cemetery') and (2) the **Graveyard Orbit** [Rabie (2021)].

- **Point Nemo** or Space Cemetery: This is an allocated section in the pacific ocean where any/all space objects, fragments, debris etc., that did not successfully burn up during their re-entry into the Earth's atmosphere, are stored/dumped. Point Nemo, taking inspiration from Jules Verne's fictional character Captain Nemo, was designated to this particular region on the planet, shown in Figure 2, as scientists framed it for being the furthest from human civilization.

Figure removed due to copyright restriction

Figure 2: The Space Cemetery: Point Nemo

- **Graveyard Orbit** or Disposal Orbit: A scientist-designated orbit that lies 300 km from the Geostationary orbit or **Geosynchronous Equatorial Orbit** (GEO), as shown in Figure 15. Retired or defunct satellites are propagated into said orbit, where these structures are left wandering in the void of space. As can be inferred, these structures are open to experiencing collisions with other similarly situated satellites or travelling celestial bodies; perpetuating the Kessler phenomenon.

As can be observed, neither of the two solutions work towards addressing the current amount of continuously increasing space debris; leaving the objects in space more susceptible to fragmentation. The action of allowing incompletely burned space junk re-entering the Earth's atmosphere provides a different set of problems. An instance of such was recorded in a news article when a "lightweight fragment of a charred woven material" struck a woman in Turkey, Oklahoma in 1997 [Wall et al. (2013)]. Due to fortunate circumstances, the civilian suffered no injuries from the event; however, it indicates that falling space debris may be received outside the jurisdiction of Point Nemo. Another recent example of accumulated space debris posing a threat to future space endeavours was the incident recorded on 5th March 2022. 3-tonnes of bundled space junk, travelling at the speed of 9300 kph, collided with the Moon's surface resulting in a crater 20 meters in diameter [Dunn (2022)]. In order to better gain perspective, scientists have stated that even a fragment of that mass is capable of causing irreversible damage to the ISS.

Exploring a different GAP opportunity, the two current means of disposal/containment do not take into consider the potential behind taking advantage recycling the debris suspended in space. On the lunar surface alone, there is an estimated amount of 150-tonnes of aluminium. The value, estimated by *Orbit Recycling*, of recycled aluminium is said to be worth \$150,000 per kilogram [SUT (2022)]; capable of covering more than the inter-celestial transport costs. The overall worth of designing concepts to address the space debris issue can be visualized when inspecting the expected growth rate in this field: where the space debris monitoring and removal market is estimated to grow from being worth *USD 942.3 million* in 2022 to *USD 1,527.7 million* by 2029 [B (2021)]. As such, it can stated that the GAP behind the problem statement lies in the current inability to address or benefit from the accumulated space junk within Earth's orbits.

4.3 Thesis Project Scope

Establishing the base layer, this thesis works under the assumption of hypothetical providing a concept or solution that could be implemented within the Low Earth Orbit (LEO). Within the duration of this thesis project, the following list of objects were selected as the points of interest to successfully achieve said proposed solution:

- **Developing Debris Identification System:** The primary aim of this thesis is to test to how viable image processing techniques are, when presented with the task to identify debris within an array of images or video. As such, the thesis is set to work on two different phases of object in image recognition. Firstly, in order to test the baseline, a python script that can determine debris in a pre-loaded video through the application of filters, kernels and contours shall be evaluated. Having established a point of comparison, the thesis aims to verify the potential of machine learning (ML) in a debris identification scene. The ML algorithm was planned to work with a comparatively larger data set than its predecessor, whilst being equipped with the ability to distinguish between four different types of objects: Astronauts, Debris, Satellites, and Extras (a label encompassing other space entities).
- **Design Modelling:** The secondary aim of this thesis is to construct a hypothetical model for the WOMBAT. This shall be achieved through the 3D modelling software of Autodesk Inventor and shall be subsequently animated to depict the sequence of actions/steps the robot would perform to successfully capture and store the target.
- **Investigating Political Market & Potential Disputes/Interference:** The tertiary aim of this thesis involves conducting research understanding the potential political issues with the introduction of such a product into the market. This involves breaking down the market reception, scrutinizing the limitations of the space maritime law and examining different country-based regulations that might need to be taken into consideration before such a product release.

5 Literature Review

This chapter of the thesis examines and scrutinizes the gathered published materials/journals pertaining to space robotics and debris identification/removal. It is important to state that there was a distinct lack of access to work conducted that incorporated both the previously iterated elements. As such, the following literature is composed of journals that either intersect or run in parallel to the scope of this project; providing data/information that could work in tandem if the project were completely restarted with a different methodology and starting resources/equipment.

The following literature review has been broken into two factions: potential robotic designs/structures and the current existing means of debris identification, in order to better collate the different works.

5.1 Investigating Potential Robotic Designs & Structures

From the available literature, an 'out of the box' concept to address the thesis's hypothetical robot could be garnered through considering Parness et al. (2017)'s introduction of a four-limbed robot. The LEMUR 3, an amalgamation of biology and an acronym standing for 'Limbed Excursion Mechanical Utility Robots', is considered the "most versatile platform" within the LEMUR series; boasting a 7 degrees of freedom (DOF) per limb approach. Pal et al. (2020)'s breakdown behind the performance of robotic manipulator (with 6 degrees of freedom) in space only accentuates the LEMUR 3's ability to perform with 7 DOF.

Pal et al. (2020)'s work more so works towards identifying the most optimal control system behind the said robotic manipulation, where they compare 4 different types: Proportional-Integral-Derivation (PID), Linear-Quadratic-Regulator (LQR), Fuzzy Logic Controller (FLC) and Sliding Mode Control (SMC). The four systems were compared based on response/time-based variables such as delay time, settling time, and peak overshoot for step input. From the perspective of time, Pal et al. (2020) recorded better results from the FLC controller; however, from a dynamics standpoint, a PID controller proved to be the better overall controller as it was more capable of dealing with non-linearity in the simulations. The PID

controller is comparatively more simple and robust in nature. The benefits from Pal et al. (2020)'s work is determining how viable a 6 DOF control system is, were the scope of the project expanded in the future. However, given that all experimental data was simulated, there will be extraneous factors when working towards developing a physical prototype that is expected to perform more than just simple movements.

To reiterate: Pal et al. (2020)'s experiments provide more of an understanding behind the applicability of Parness et al. (2017)'s LEMUR 3. The LEMUR 3 was designed with the following objectives: (1) "[crawl] across the exterior of the International Space Station" and (2) "[climb] vertical cliffs and [traverse] cave ceilings on the Moon and Mars". As as can be observed, the LEMUR 3 was built for a different purpose rather than space debris identification, capture and/or removal. However, the design could be modified to consider more of an observation role from the aforementioned ISS's exterior; as even having locational and potentially traversal knowledge would greatly benefit other space debris management endeavours. One such endeavour could include working with Srikrishnan et al. (2015)'s **Single semi-Autonomous Satellite Tracking ROBOT** or SASTROBOT.

Exploring a different avenue of reducing space debris, the team at the Institute of Aeronautical Engineering have a conceptualized a means to taking advantage of the numerous dead satellites in LEO [Srikrishnan et al. (2015)]. The SASTROBOT is designed to seek out defunct satellites and attach a deorbiting rocket onto the structure. The selected targets for said attachment is depending upon the Area Mass Ratio (AMR) of the satellite and all control is directly placed to an assigned ground station; who is responsible for safely bringing back the dead satellite through a stable de-orbit. Given that Srikrishnan et al. (2015)'s mission aims to carry 20 deorbiting rockets and 'capture' 18 defunct satellite per run, the overall number of launches and mission cost would be within manageable limits. Before re-launching the SASTROBOT for subsequent runs, it could be equipped further "mission-based segments" to make some defunct satellites more retrievable. However, a key factor that draws a line between Srikrishnan et al. (2015) *vs* this project is its scope. Srikrishnan et al. (2015) is a mission on a grander scale to larger, full-sized defunct satellites whereas this thesis or WOMBAT is aimed at

addressing the smaller fragments or remains distributed in space, due to collisions from such said defunct sources. As such, before specifically looking at features that we would like to equip the WOMBAT with, a better understanding of the space communities' active concepts of debris removal was needed and was achieved through Singh et al. (2020)'s astute breakdown. This type of emphasis relate back to the compaction of the thesis's WOMBAT.

Amongst the plethora of active removal concepts, Singh et al. (2020) starts out with evaluating the Drag Augmentation System, highlighted as one of the more capable concepts. This method aims to increase the drag of the object (i.e., debris) by "enhancing the area towards a mass ratio of the objects"; theoretically aiming to throw some debris through a small density. This concept was further broken down into three methods: foam-based, fiber-based and inflated. Following a similar format, these methods aim to launch chaser satellites that cover the targeted debris/objects with foam/fiber/inflated ball respectively. Other methods (such as the slingshot method, laser satellite method, ion beam shepherd method) discussed in Singh et al. (2020)'s work display over-arching themes involving displacing the debris such that its movement allow the debris to either be shifted towards the graveyard orbit or into the Earth's atmosphere. Whilst this research provided means on displaying debris, it continues to dependent upon the two current means of space debris management: the graveyard orbit and/or Point Nemo. Singh et al. (2020) does not scrutinize the potential applicability for more 'traditional' capture methods such as robotic grippers, nets or harpoons and in order to do this, we look at an outlier case - Lv et al. (2022) and their application of all three previously mentioned capture types simultaneously.

Lv et al. (2022)'s paper conceptually researches a platform that integrates three means of capture, namely, a robotic gripper, a harpoon, and a tethered net, and examines the complementary advantages of applying them simultaneously. Initially, the capture is achieved through locking onto the debris through the harpoon. The tethered net is then overcast to stabilize the tumbling speed of the debris; having collided with the harpoon. The tether is then reeled in towards the system and during this action, the robotic arm then collects and internally stores the debris. This journal provides an exemplary hypothetical working of such an concurrent

task. Each of these manual capture methods were thus taken into consideration for the WOMBAT's theoretical design. However, given that Lv et al. (2022)'s work is also hypothetical in nature, whilst the method is effective in theory, the project does not consider the required resources, costs, control systems etc., that would need to be accounted for; factors that could potentially drive away investors from supporting a potentially expensive outcome.

Having looked into the various variables behind the construction of a space robot, the following section of the literature review targets the different methods used or conceptualized to locate/identify space debris.

5.2 Investigating Potential Means of Space Debris Identification

An aspect when working towards a hypothetical space debris robot, apart from investigating for its physical structure or features, is the debris detection/identification; the core component of this thesis. Given that the scope of the initial thesis proposal included the possibility of developing a machine learning algorithm, we shift to a similarly intended work by Perez et al. (2021). Given that the scope of their work, Perez et al. (2021) work with two different types of images for their system: (1) synthetic images and (2) representative images generated by pre-built test facility. The algorithm was constructed to achieve the following characteristics: (1) distinguish between satellites and non-satellites, (2) assess the stability of the satellite by estimating its tumbling rate, solar panel orientation, change in pose, etc., and (3) determine the material composition of the satellite. Having established the requirements, the explored artificial learning was categorized to two varieties: classic machine learning algorithms and deep learning/neural networks. Upon tackling the task, the Perez et al. (2021) team achieved relative success to support their 'proof of concept'. However, results were not achieved without drawback for each system as the Machine Learning based methods were observed to be position-sensitive; depending on the pixel values. Perez et al. (2021)'s Deep Learning, or Convolutional Neural Networks, did not exhibit this ability due to their greater visual processing capabilities; enabling this system to more distinctly identify visual features within the images. As such, this journal paved a road to-

wards supporting this thesis's initial scope. However, in order to further expand our options, the research dove towards scrutinizing Mehrholz et al. (2002)'s work and their application of radar beams.

The following journal was more an inspection behind radar beams and could not be considered within the thesis due to the lack of proper resources. This was accentuated by the journal speaking of implementing the FGAN Tracking and Imaging Radar (TIRA): "a 34m parabolic antenna, a narrow-band mono-pulse L-band tracking radar, and a high resolution Ku-band imaging radar". If this facility could work in tandem with previously reviewed work of Srikrishnan et al. (2015), the project overall would gain data and the ability to do the following:

- search and track space objects
- characterise the current space-debris environment
- validate space-debris systems/models
- keep track of space fragments or debris re-entering the atmosphere
- space objects imaging (further showing potential in work with research by Perez et al. (2021))

As such, despite the work presented by Mehrholz et al. (2002) being outside the scope of the project, this investigation opens other avenues of research that could potentially be complemented with such information. On a similar note, the Western Australian team of Tingay et al. (2013) showcase the application of the Murchison Widefield Array (MWA), a new low-frequency interferometric radio telescope, for space debris endeavours.

The paper explores the application and effectiveness of MWA for general **Space Situational Awareness (SSA)**. For their setup, the MWA acts as the "receiving element in a bi-static radar configuration" whereas the FM broadcast stations play the role of transmitters. The waves transmitted propagated into space reflect off debris situated in LEO, which are then collected at the receiver, the MWA. With this data processed, the MWA is then capable of the following: (1) detecting multiple fragments of space debris (detecting an average of 10 pieces at any given

time), (2) "image their positions on the sky as a function of time", and (3) calculate orbit parameters through the available tracking data. A factor, amongst others, that weakened the overall reliability of the results is that the project conducted its studies under one assumption that the sky does not change over the course of the observation. The assumption loses validity due to the model working with space debris' rapidly changing angular motion relative to the backdrop of images being either the sky or other celestial bodies. Tingay et al. (2013)'s paper is expected show better and reliable outputs were all the weaknesses addressed and similar to Mehrholz et al. (2002), shows better applicability when working with other aforementioned projects such as Perez et al. (2021), Srikrishnan et al. (2015), Lv et al. (2022), etc.

5.3 Section Overview

Formally concluding this chapter of the thesis, the multiple avenues of research and investigation initially displayed the lack of pre-existing data and resources that the WOMBAT could be built upon. Amongst the plethora of journals scrutinized, some concepts such as using image processing and machine learning were supported in the field of debris identification, despite the thesis's scope being smaller in scale. On a similar note, it was better recognized that hypothetically 3D designing the WOMBAT's structure with more 'manual' means of capture and navigation posed as being more 'realistic'; if the project was ever to be realised.

6 Methodology

This chapter of the thesis breaks down the three different engineering goals that were conducted over the duration of this project: (1) kernel-based image processing, (2) machine learning algorithm, and (3) 3D modelling a hypothetical robot. All code implemented for tasks (1)(2) shall be appropriately credited and provided within the appendices of this report. Comparatively, goal (3) is more briefly expressed due to the nature of the task and a definitive steps to expand upon; however, has a more prominent role in the 'Discussion' section of the report.

6.1 Equipment Configurations

The section aims to briefly provide an overview regarding the access to hardware and software that enabled progress with the scope of the thesis.

6.1.1 Hardware

All three engineering tasks (1), (2), and (3) were completed on a personal laptop: Acer Predator Helios 300 equipped with an Intel(R) Core(TM) i7-8750H processor with 16.0 GB RAM. The device is comparatively more capable than standard personal devices, but it did require certain accommodations during the machine learning phase of the thesis.

6.1.2 Software

All coding constructed and implemented for aims (1) and (2) were completed through Python programming. Aim (1) was facilitated through the web-based interactive interface - Jupyter Lab. On a similar note, Aim (2) was completed through accessing Jupyter notebooks; where Jupyter notebooks are simply a smaller subset of the greater Lab interface. Aim (3) was undertaken through working with the 3D modelling capabilities of Autodesk Inventor.

6.2 Primary Goal: Part 1 - Kernel-based Image Processing

Prior to analysing the code implemented in this section, it is vital to acknowledge the assistance from the sources behind each of the resources used. Firstly, the code support, inspiration and assistance is credited to Simran Suresh and her publicly available github repository [Suresh (2022)]. Other resources utilised within the execution of the code are snippets from the movie "Gravity" starring Sandra Bullock and George Clooney. These movie clips were used as the material upon the program overlays and detects said debris.

NOTE: The code shall be broken down into two individual sections in order to better track its gradual functionality. A similar format shall be followed when providing said code within this report's appendices.

6.2.1 Listing 1: Initializations, Capturing the video and Setting up the kernels

Listing 1 begins with the importing of the two vital libraries required for this program: *cv2* and *numpy*. *cv2*, an opencv-python importing module, was imported as it allows the program to gain access to a multitude of image processing functions and packages; this library shall be accessed through the assigned key word *cv2* in later steps. *numpy*, an array-based package, was imported into the code as it provides functions that work in tandem with multiple mathematical expressions, specifically looking at matrices; as the process of image processing works with identity matrices, convolution, etc. These shall be addressed through the keyword of *cv2*.

Following this, it is necessary to state an assumption that the overall system works upon: 'all regions in the video/film excluding debris is considered black'. As such, the two array variables establish the lower and upper boundaries through the respective Hue Saturation Value (HSV) configurations for the colours of black ([0, 0, 0]) and white ([0, 0, 255]); the reason behind the application of HSV colour coding shall be provided in later steps. Having set these boundaries, the *cv2* library is accessed and applied to capture the .mp4 snippet of "Gravity" in the variable *cam*. Following this, for later purposes of 'outlining', two kernels (opening

and closing) are set up in the following form: `np.ones((5, 5))` and `np.ones((20, 20))` respectively generating identity matrices [rows, columns] of [5 x 5] and [20 x 20] with the value of 1 in each cell.

6.2.2 Listing 2: Resizing, Color conversion, Mask generation and manipulation, Contouring, and Displaying final output

Listing 2 begins with setting up the *while* loop in which the rest of the remaining code is executed. Within this loop, the first line of instruction stores two different variables from reading the video's frames. The variable *img* stores an image array vector corresponding to the default frames per second, where if the frame is available, returns a 'true' value that is stored within *arg*; this boolean value shall later work in tandem with a while loop. Following this, the captured frames are resized to the dimensions of 1366 x 768, a default resolution used for standard computer and laptop screens - a setting that saves memory, reduces the file size all whilst retaining the quality of the image. Promptly following this, the program is instructed to convert the captured frames from a Blue-Green-Red (BGR) to High-Saturation-Value, matching its type to previously mentioned upper and lower boundaries. Other factors that require this conversion include that the BGR represents color intensity/luminescence, making it difficult to separate colours within it. On the contrary, HSV separates the colour information from image luminescence.

The following three lines of code generate and morph the necessary mask that enables the program to perform 'accurate' edge detection in later phases; the previously constructed filtering kernels are implemented in tandem with the morphological functions to perform the required dilation and erosion. Having 'morphed' the image, the contours and hierarchy are returned as values from the application of the *findContours* function. The detection of colours are stored in form of vector points and returned as the data pertaining to 'contours' for the variable *conts*. The variable *h*, representing hierarchy, is returned output vector that stores essential information related to the typology of the image. With the necessary information available, this listing of the code concludes with drawing the contours onto the video frames with set line colour (white) and labelling font of hershey.

Following this, for the 'length' of the determined contours, bounding rectangles were constructed and overlapped over the discerned 'debris' or objects. The previously set text font was utilised as a trait for the numbering of detected objects within the video. Thus, having built the necessary overlays for various frames of the video within the *while* loop, the final output is portrayed in a new window. The user was also provided the ability to collapse the output window at any given time with the keyword '*q*'; breaking free of the infinite *while* loop.

6.3 Concluding Comments

This covers the entirety of the code implemented to achieve the kernel-based image processing program for debris detection. All obtained results shall be scrutinized within the **Results** and **Discussion** sections of this report. Having completed, the project makes strides to test the viability and accuracy of machine learning in such a scenario.

6.4 Primary Goal: Part 2 - Machine Learning Algorithm

On a similar note to the previous section, prior to analysing the code implemented in this section, it is vital to acknowledge the assistance from the utilised resource. The code support, inspiration and assistance is credited to Nicholas Renotte and his publicly available github repository [Renotte (2020)].

NOTE: There are two phases/sets of code encompassing the machine learning algorithm: (1) Image Collection and (2) Training and Detection. These phases shall be further broken down into multiple listings to better align this information with the structure of this report.

6.4.1 Virtual Environment

The project initiates with the creation of a separate virtual environment, named *tfod*, in order to have complete control over every aspect of the algorithm. This step was taken as a virtual environment allows the user to keep track of every package subsequently installed while controlling their respective version and/or updates. This was achieved through the running the following command in the

anaconda environment: `python -m venv tfod`. The only drawback with this initial setup is ensuring that there is excessive use of third-party modules that would a pip install.

6.4.2 Listing 3: Initializations and Image Collection

As previously established, since a separate virtual environment was constructed for this machine learning algorithm, the code in listing 3 starts with the installation of the necessary packages and files from the `opencv`, or open computer vision, library/dependency. Following a similar note as the kernel-based program, the code subsequently imports the `cv2` package along with `uuid` and `os`. The `uuid` package enables a robust way to generate unique identifiers within the python environment, whereas the `os` package offers the ability to establish interaction between the operating system and the algorithm/user.

The next step of the program involves creating an array with the different type of objects that you required the algorithm to later detect. As such, following the scope of this thesis results in the following four labels/'objects' as being the focus of interest: (1) debris, (2) astronaut, (3) satellite and (4) extras (this category refers to any other lower priority space entities such as asteroids etc). It is important to take note of these label 'names' when navigating through the labelling section in the next listing as it was observed that the algorithm was case sensitive in this regard. Having identified the desired categories, the code constructs the following path into the virtual environment folder through the `os.path.join()` function: Tensorflow > workspace > images > collectedimages. The following *if* statement acts as a double-check ensuring that the previous images path has been constructed; if the condition is met, it re-executes and constructs the path. This listing concludes with a *for* loop that adjoins four folders (subsequently named based on the labels) onto the same path.

This phase of the code was concluded through surfing the internet to collect high-quality synthetic images (due to a lack of access to real images of debris etc.) that fall under the four categories and are respectively stored in their generated folders.

6.4.3 Listing 4: Image Labelling

Listing 4 starts with the need to pip install specific upgrades for *pyqt5lxml*, based on the requirement for using Tzutalin (2016)'s labelling tool. Another folder, named "labeling", was then added into the path or directory of the original "TensorFlow" environment folder. Once verifying that the previously stated file path was successfully integrated, all folders, files and packages from Tzutalin (2016)'s repository were cloned into this "labeling" folder through the `!gitclone` "URL" command. These files were then subsequently installed and the labelling tool/software, 'python labeling.py' was executed.

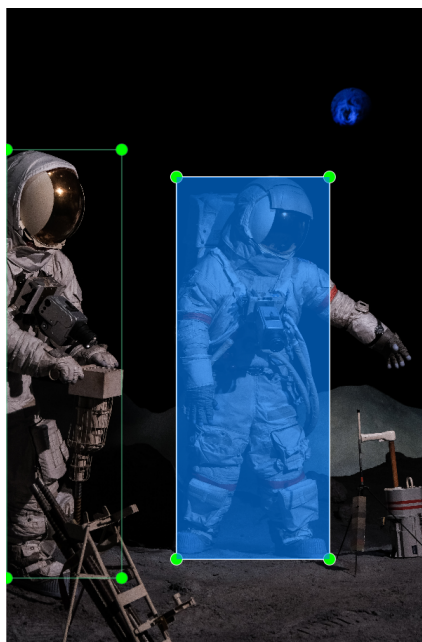


Figure 3: Tzutalin's Labelling Software/Tool

Figure 3 displays how the labelling tool activates allows the user to manually draw contours/boxes over multiple objects in one image and label into one of the four categories. This process was repeated for over 50 images for each type of to-be-detected objects (totalling at $4 * 50 = 200$ labelled images). These labelled images were then readily accessible for the next major phase of the machine learning algorithm.

6.4.4 Listing 5: Part 1 - Setting Up the Training and Detection Algorithm

Given that the training and testing phase of the project was initiated in a different notebook, listing 5 starts with the importing of the *os* package. The following three lines of instruction address an external pre-trained model named 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8' that refers to one of many models available from the tensorflow 2 detection model zoo Tensorflow (2017). With an average speed of 22 ms and mean average precision of 22.2 mAP, this model was selected in particular for the following list of factors:

- **Limitation:** Was most stable compared to other models when implemented on personal hardware.
- **Versatility:** obtained results or outputs can be replicated on less capable device such as Raspberry Pi or mobile phones. This provides more flexibility if the project were to reach an official robotic design phase.
- **Automated Processing:** This pre-trained model performs the necessary pre- and post- processing required by models when compressing and decompressing input images/videos from their original resolution to 320x320.
- **Image Augmentation:** The model automatically performs additional manoeuvres such as darkening the image, altering the orientation, etc., and thus accounts for more possibilities than just initially provided list of images; with the intent to improve model performance.

As such, for the reasons above, the algorithm was given access to the URL that enabled to interact with the required github repository. Following this, variables to store the necessary TensorFlow records (data records that are required for the test/train phase) and the label map (a text file that maps the labels to their respective IDs/identifiers).

The next series of instructions stored within the *paths* dictionary utilises the *os* package to create the necessary file paths within the constructed virtual environment. Amongst the list of paths built, any data acquired from the model zoo (the SSD_mobilenet model) is allocated to the path constructed for *Pretrained Model Path*.

Other file paths of interest include the '*Checkpoint Path*' and the '*Protoc Path*' - each of which shall be explored at later stages of the code.

The *files* dictionary performs a set of instructions to configure the required settings/paths for each of three depicted variables: *Pipeline Config*, *TF Record Script* and *Label Map*. The role of *Pipeline Config* and the *pipeline* file shall be discussed in later sections of the code. Following this, all constructed file paths are described above are officially added into the directory within the *for* loop.

6.4.5 Listing 6: Part 2 - Setting Up the Training and Detection Algorithm (Continued)

Listing 6 begins with the installation and importing of the *wget* library that provides the feature to pull down the required pre-trained model from the TensorFlow 2 detection model zoo. The next two lines of instruction verify for an existing path and clone an essential object detection repository from the TensorFlow model garden into this directory.

The next subsection of code refers back to the previously stated *ProtocPath* as it installs all required protocol buffers for the algorithm. These protocol buffers are necessary as they enable the compiler to interact with corresponding model data; alongside integrating these buffers, the code installs the official object detection API. However, before progressing with future steps, it is important to ensure all necessary files have been installed/imported for the algorithm. As a means to double-check the work, a verification script is run to perform a background check where it revealed the required lines of instructions (the "Missing packages and libraries" subsection) necessary; leading to their inclusion. Lastly, to conclude this listing/section of code, the SSD_mobnet pre-trained model is downloaded into its designated file path.

6.4.6 Listing 7: Part 3 - Setting Up the Training and Detection Algorithm (Continued)

Listing 7 starts the subsection of code with the construction of the label map. The labels from the 'Image Collection' phase of the projects are replicated (as these

labels are case-sensitive) and assigned in a map with designated IDs; this label map is later utilised by the pipeline file during the testing phase of the algorithm. In order to gain access to an open source data analysis and manipulation tool, the files and packages from library *pandas* were installed. From Renotte (2020)'s repository, the script that is capable of generating TF records or data sets is cloned into its respective file path. This script is then subsequently implemented to build the two 'train' and 'test' data sets (or TensorFlow 'Records'). The 'train' data set is constructed based on a 100 labelled images comprised of the four object types (25 labelled images for each category). On the other hand, the 'test' record works with other 100 un-labelled images as it is the data set that verify the algorithm's ability to detect said objects.

6.4.7 Listing 8: Part 4 - Setting Up the Training and Detection Algorithm (Continued)

This subsection starts with the copying the previously developed configuration files into the training file path, followed by which, a series of instructions import the necessary files and packages from the libraries pertaining to transfer learning and pipeline files.

Transfer learning refers to the concept where the pre-trained model uses past data and information on the current training and testing data sets to improve the performance of the model. Given that our machine learning algorithm revolves around utilising the SSD_mobnet model, the remaining lines of instructions in listing 8 configure pipeline file with necessary settings. The algorithm's pipeline configuration file controls the architecture and defines what the overall model looks like. This model definition enables this algorithms to utilise the previously constructed label map and two of the SSD_mobnet's features: (1) Automated Processing and (2) Image Augmentation. Thus, with all the steps achieved upto this listing, the following sections of the program can implement the training, evaluation and testing phases of this machine learning algorithm.

6.4.8 Listing 9: Part 5 - Running the Training Phase, Evaluation Phase and Setting up Checkpoints

Listing 9 initiates with running the training script to build the machine learning algorithm's data set based on the previously iterated 100 labelled images. The printed *command* line is executed within the Anaconda terminal for 6000 steps. Previously, when the number of training images fed to the algorithm were 50, the training phase had been executed for 20000 steps for reasonable results during the testing phase. However, when the number of images was incremented to 100, running the training phase for 6000 steps yielded equivalent results; and as such is presented in the code.

Following this, in order to measure the effectiveness of this ML algorithm, an evaluation command is inputted into the Anaconda terminal. This data provided a visual depiction of the algorithm's capabilities and shall be scrutinized within the 'Discussion' section of the report.

Prior to running the testing phase of the algorithm, it was necessary to import and re-import (for safety measure) certain files and packages that would assist the construction of checkpoints. The remaining lines of instructions in this listing, working in tandem with the architecture defined by the pipeline config file, allowing the algorithm to work in checkpoints. Checkpoints allow the algorithm to continue training from their previous endpoint. The run/iteration where the training was conducted for 20000 steps, was initially broken down into multiple checkpoints: (1) 2000 steps, (2) 4000 steps, (3) 10000 steps and then led up to (4) 20000 steps. This allows the user to potentially evaluate the viability of the ML algorithm under different number of steps if required.

6.4.9 Listing 10: Part 6 - Running the Testing Phase

Listing 10 sets up the required packages and libraries prior to running the following lines of code. The *ImagePath* variable is then executed through the *os.path.join()* to collect the image to be tested through the third input parameters to verify whether algorithm identifies its contents. Following the remainder of the instructions, as per the guidelines of Renotte (2020), perform the steps required to cal-

culate the detections. Upon determining the objects detection, the code applies boxes around the target whilst providing information on the derived confidence metric simultaneously. The confidence metric refers to the numerically representing how accurately the algorithm believes it has recognized the object; presented in percentages. The closure of the listing apply these visual contours and metric data to display to the user through the *plt.imshow()* and *plt.show()* commands.

6.4.10 Concluding Comments

This concludes the primary software goals that were expected to be completed as part of the scope of this thesis project. For access to all code discussed in the above section, please refer to the listings provided within the report's appendices. The effectiveness and viability of these developed software shall be evaluated in the upcoming sections of the thesis.

6.5 Secondary Goal: 3D Modelling

Compared to other goals of this thesis, this secondary aim does not comprise of numerous engineering tasks or steps to discuss. The methodology behind the construction of this thesis's robot WOMBAT shall be broken down into sections: (1) Design Phase and (2) Engineering Phase. The design phase shall discuss the decisions that were considered behind the hypothetical functioning of the WOMBAT. On the other hand, the engineering phase shall provide a succinct overview of the tools utilised to construct the 3D model.

6.5.1 WOMBAT: Design Phase

Given that this thesis project is based in Australian, this thesis's supervisor, Mr. Pepe' Joseph Velasquez, provided the concept/idea of modelling the robot following a possible bio-mimicry of the mammal, wombat. This was used as the concept/structural baseline when working towards designing the robotic model.

The objective or priority target of this thesis was to design a robot capable of collecting and either recycling/disposing of space debris $< 10 \text{ cm}$ in dimension. Using this as a design parameter, the decision to make the size of the WOMBAT

be equivalent to that of a standard mini-van (where the typical dimensions are: length x breadth x height = 2.4 m x 1.72 m x 1.94 m). These dimensions were proportionally reflected (in mm) when 3D modelling the WOMBAT in Autodesk Inventor. Having approximated the general size of the robot, it was necessary to contemplate potentially desirable features.

The interior of the robot's structure was designed to house 20 to 30 kg of space debris at max capacity. If design and resources enable to do so, the robot would be designed with the ability to store its contents in a cube-shaped compartment. The decision was made in tandem with equipping the WOMBAT with a square-shaped cavity to potential dispose debris by dropping and letting the space junk burn up in the Earth's atmosphere. The square-shaped cavity is to mimic the wombat, being the only mammal on the planet to excretes in cubes. However, if resources and propulsion allow the WOMBAT to instead recycle the space debris, with plans to regularly travel back and forth with the International Space Station (ISS) are being taken into consideration. Due to the limitation on time frame, this project was unable to conduct research into the specific resources required to realistically build this robot and/or design the type of equipped propulsion methods; potentially being an extended project for a future researcher. Given that the objective of this robot was firstly seize any space debris, the type of desired capturing mechanism was evaluated.

From the various literature considered for this aspect of thesis project, the potential options for capture mechanisms were narrowed to three 'manual' modes of retrieval: (1) Robotic Claw Capture, (2) Net Capture, and (3) Harpoon Capture. In order to further assist the decision making, each of these captures' advantages and disadvantages shall now be evaluated.

- **Robotic Claw Capture**

- **Advantages:**

- * The design allows more flexibility due to being able to incorporate limbs with 6 degrees of freedom (DOF) into its movement patterns.
 - * Managed by a control system, providing overall stability during the

capture process.

- * Low levels of risk of causing fragmentation of debris upon contact.
- * Greater versatility in regards to greater sizes of debris capable of being captured.

– **Disadvantages:**

- * More resources and expenditure required for this type of capture (Comparatively the most expensive option amongst the three types).
- * More potential sources of error or malfunctions with this type of product.
- * More energy intensive for the WOMBAT to sustain.
- * Equipped better to deal space debris with bigger dimensions; does not appeal to the robot's need of retrieving junk < **10 cm**.

• **Net Capture**

– **Advantages:**

- * The size of net determines the type of debris capable of being retrieved; depicting high levels of versatility.
- * Does not require resources invested into designing control systems/programming.
- * Amongst the three, theorized to be the least reliant on the travelling speed of the target debris.
- * Low levels of risk of causing fragmentation of debris upon contact.

– **Disadvantages:**

- * The net might need to be modified towards a mesh or sheet design to reduce the risk of debris escaping the material.
- * Net has to be created from a highly durable material in order to intercept debris speeds; potentially increasing overall expense.

- * Information regarding the target has to be determined prior to applying this technique as the dimensions of the net/sheet define the target.

- **Harpoon Capture**

- **Advantages:**

- * The nature of the capture prevents the debris from escaping during the retrieval process.
 - * Comparatively less intensive as it expends the bigger proportion of its energy during the harpoon launch.
 - * Does not require the same level of resources as the robotic claw capture.

- **Disadvantages:**

- * Low levels of risk of causing fragmentation of debris upon contact.
 - * Very reliant on calculations and timing to intercept space debris travelling at considerable speeds.
 - * Cannot be utilised for space debris very small in constitution; acting as a factor derailing it from the WOMBAT's design.

This concludes the general analytical breakdown behind the design decisions made for the WOMBAT and all selected characteristics shall be examined in the "Results" and "Discussion" sections of the report. The following section shall succinctly describe the steps behind 3D modelling the robot.

6.5.2 WOMBAT: Engineering Phase

All modelling completed within this phase of the project was conducted with the same hardware implemented for the project's primary software goals. The 3D model was designed with the assistance of the computer-aided design application Autodesk Inventor, and the same software was utilised to create the working animation of the WOMBAT.

6.6 Section Overview

With this, the chapter of the thesis scrutinizing the methodology behind the thesis has concluded. The recorded results/outputs obtained from the kernel-based image processing program and the machine learning algorithm shall be examined in the following sections of the report. On a similar note, all decisions made towards the WOMBAT's design and the final design outline shall be disclosed in the same manner.

7 Results

This chapter of the thesis provides a brief underline of the various results obtained from applying the steps discussed in this report's methodology. Following a similar trend, all recorded results in this thesis shall be divided under three subsections: (1) Kernel-based Image Processing, (2) Machine Learning algorithm, and (3) 3D Modelling & Animation.

NOTE: This section will be brief in nature as all results' evaluation shall be situated under the "Discussion" section and role of this section is portray the outcomes to the reader.

7.1 Kernel-based Image Processing

As the scenario presents the threat of space debris that this thesis project is written to convey, two separate clips from the movie "Gravity" were utilised to test the program's active debris detection. Due to the inability to integrate moving images or videos into this thesis report, a working frame from each output shall be provided below to depict the code in action.

Figure removed due to copyright restriction

Figure removed due to copyright restriction

(a) Scene 1 - Captured Action Frame

(b) Scene 2 - Captured Action Frame

Figure 4: Kernel-based Image Processing Outputs

As can be inferred from Figures 4a and 4b, the program counts and places contours/bounding boxes around every object that its algorithm deems falls under space debris. These results shall be evaluated in the later section of the report.

7.2 Machine Learning Algorithm

Having trained the program, the following series of images were applied to test its object recognition capabilities. As had been previously iterated as a limitation to the overall applicability of the machine learning algorithm, this case works with training and testing **synthetic** images, i.e., images that do not represent the real scenario.

Figures removed due to copyright restriction

(a) Astronaut Recognition - Image 1

(b) Astronaut Recognition - Image 2

Figure 5: Machine Learning vs Astronaut Detection + Confidence Metrics

Figures removed due to copyright restriction

(a) Extras Recognition Example

(b) Satellite Recognition Example

Figure 6: Machine Learning vs [Extras, Satellite] + Confidence Metrics

Figure removed due to copyright restriction

Figure 7: Identifying two different objects: Example 1 - Astronaut + Satellite

Figure removed due to copyright restriction

Figure 8: Identifying two different objects: Example 2 - Astronaut + Debris

Figures 5a, 5b, 6a, 6b, 7, and 8 indicate that the machine learning algorithm was capable of identifying specific regions of each image with a fairly high confidence metric. However, further analysis highlighted the weaknesses in its learning and these shall be examined in discussion.

7.3 3D Modelling & Animation

This subsection of results shall provide illustrations and images of the 3D model designed for the thesis' WOMBAT. The following series of images shall provide depictions of the following data/information: (1) WOMBAT Schematics, (2) 3D WOMBAT model, and (3) Capture Frame of WOMBAT in action. For similar reasons as the kernel-based processor, the lack of integrating videos into reports resulted in attaching a frozen frame of the robot in action.

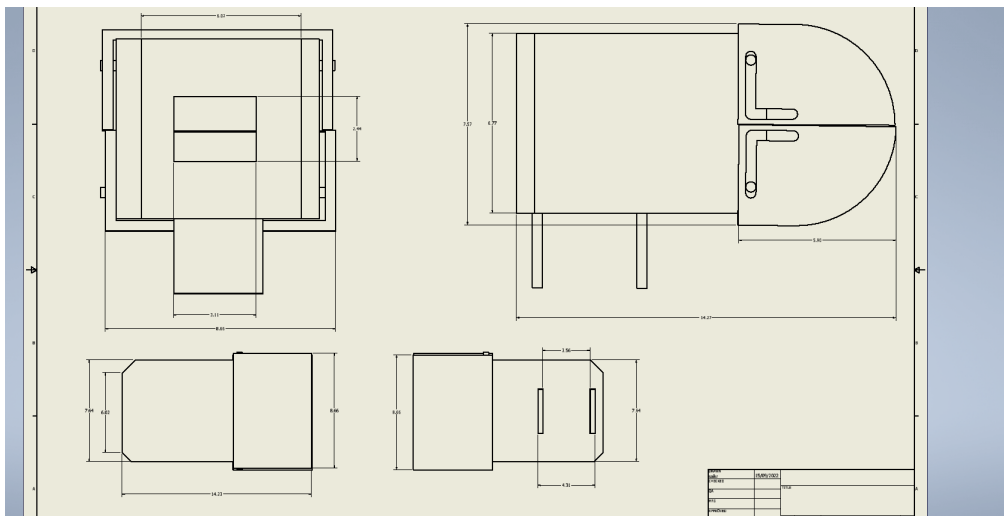
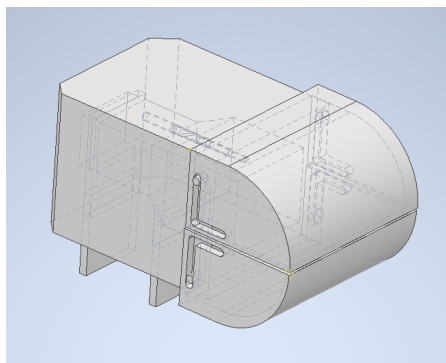
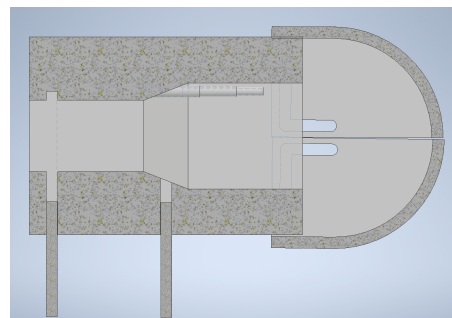


Figure 9: WOMBAT: Schematic Model



(a) WOMBAT: Full 3D Model



(b) WOMBAT: Cross-sectional View

Figure 10: WOMBAT: Full + Cross-sectional View

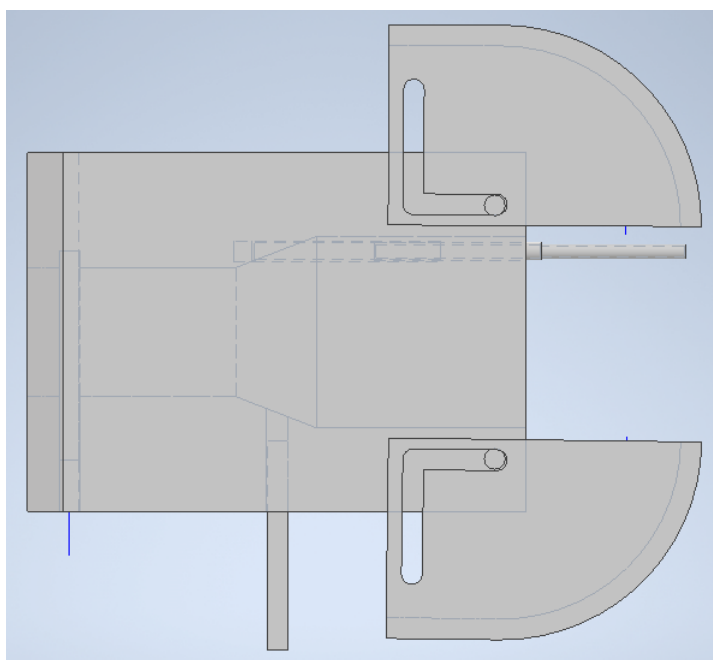


Figure 11: Animation Frame: WOMBAT in Action

7.4 Section Overview

This concludes the results obtained from the various goals achieved over the course of this thesis project. These results shall now be scrutinized in order to evaluate the overall viability of the kernel-based processor and the machine learning algorithm, alongside investigating the hypothetical features integrated into the WOMBAT's model and their subsequent drawbacks/weaknesses.

8 Discussion

Following the structural format that has been implemented in multiple areas within this report, this section shall subsequently be broken down as such: (1) Kernel-based Image Processing program, (2) Machine Learning algorithm, and (3) 3D Modelling Animation. However, in addition to these sections, the report shall also dive into a 4th section revolving around addressing the security and political issues with potentially introducing such a product into the current market.

8.1 Kernel-based Image Processing Program

Figures 4a and 4b represent the overall effectiveness when implementing the program for movie-based scenarios. As can be observed, the program successfully applies bounding boxes (contours) based on its internal calculations, whilst providing a general quantity of debris 'detected'. However, the program's inconsistencies are also visible within the same figures as it 'identifies' regions of the astronaut or the man-made structures as 'debris'. This relates back to the fundamental weakness within the development of the code as the detection is reliant on color/pixel recognition as the program works on the following baseline assumption: "all regions but debris is black in space". Whilst the accuracy of the system was improved through integration of kernels and masks, this initial condition adds a source of error that carries across the program.

The drawback with this method of object detection is that the program is very dependent on the color values (pixels) of images or videos being used as inputs. When navigating through the expanse of space, the WOMBAT is susceptible to experience varying levels of lighting; contributing to the number of inconsistencies that the algorithm has to manage. The secondary drawback of this algorithm is that it is not testing against files that are purely black + white, thus, not providing a fair representation of its ability to work efficiently in space. As such, the thesis's scope was configured to integrate and verify the viability of the next engineering task - the machine learning algorithm.

8.2 Machine Learning Algorithm

It is important to state that the number of images used for the training and testing phases changed over the duration of the thesis, where the ML algorithm underwent two different complete iterations of learning. The first learning set involves using only a total of 50 images (25 for training + 25 for testing) which required the algorithm to learn for 20000 steps to achieve satisfactory results. All results discussed in this thesis, including the outputs displayed in figures 5a, 5b, 6a, 6b, 7, and 8, were the results of the second iteration of learning. In this run, a total of 200 images were collected and fed into the algorithm (100 for training + 100 for testing). This showed an instantaneous improvement in performance as high confidence metrics were observed at 6000 steps. However, in order to appropriately assess the activity of the machine learning algorithm, it is important to interpret and extract its results in three data types: (1) Loss Curves, (2) Precision (mAP) and (3) Recall.

8.2.1 Recorded Loss Curves

Loss curves or functions is a means to "[evaluate] how well your machine learning algorithm models your featured data set" [Gupta (2022)]. There are a numerical measurement or representation of the model's ability to predict the required outcome, i.e, in this case, evaluate how good the model is at predicting whether it can identify the four designed objects: astronauts, debris, satellites and extras. Amongst the different types of loss curves that can be generated, this thesis is going to evaluate the model based on the following list of 4:

- **Classification Loss:** These loss curves have further subsets among which the most common type was selected for this thesis: Binary Cross-Entropy Loss. This loss curve measures the performance of the model whose predicted outcome lies between the probability of 0 to 1.
- **Localization Loss:** This type of loss curve represents the smooth absolute error or L1 loss between the ground truth values and predicted bounding box corrections.
- **Regularization Loss:** The act or process of regularization refers to the

background modifications that the algorithm performs to alter the weight of certain values to favor a more simple prediction. This is typically performed to prevent events such over-fitting to occur in neural networks.

- **Total Loss:** This curve simply represents the quality of the model's prediction over its course/learning.

Having provided a general overview regarding the four types of loss curves that are going to be analysed, the following Figure 12 provides the data representing this thesis's machine learning algorithm:

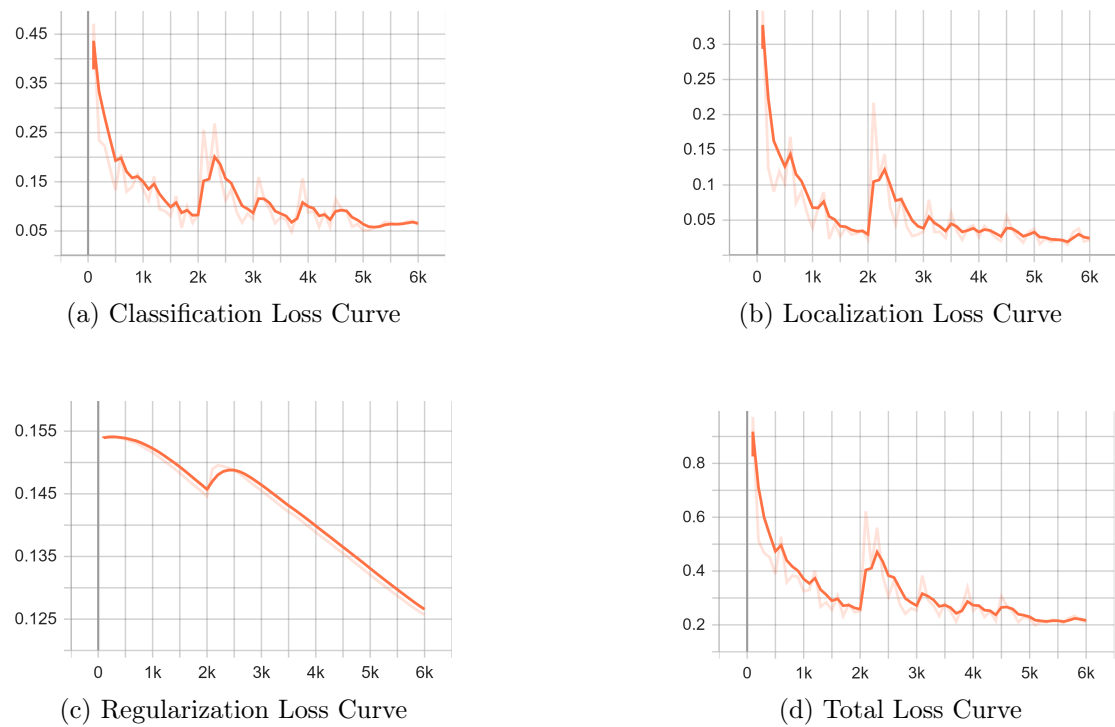


Figure 12: The 4 types of Loss Curves

From Figures 12a, 12b, 12c, and 12d, it can be observed that machine learning algorithm faces a equivalent unknown discrepancy between steps 2000 - 3000. However, given that the ideal loss curve for a machine learning model is expected to approach the value of 0 as it trains/tests, the overall results indicate that the constructed thesis model is fairly precise with its predictions.

8.2.2 Recorded Precision - Mean Average Precision (mAP)

The following graph represents the proportion of positive identifications that were actually correctly identified over time. The precision (mAP) calculated is defined by the following equation/expression: $\frac{TP}{TP+FP}$, where TP and FP respectively refer to recorded True Positives and False Positives.

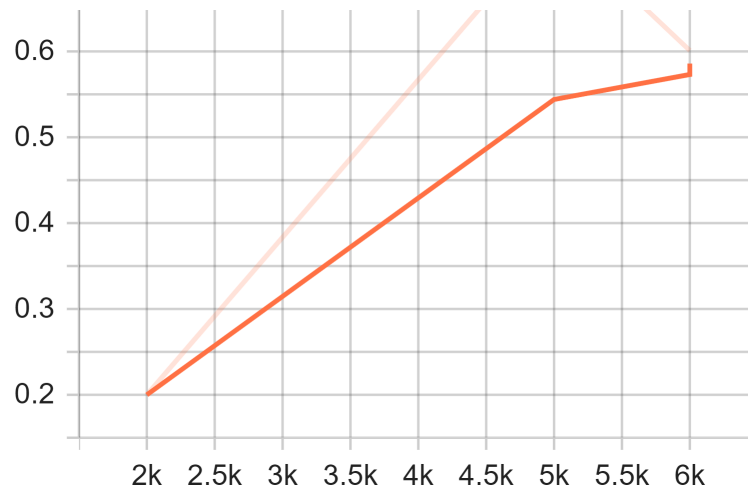


Figure 13: Precision Curve over 6000 steps (mAP)

The graph indicates that as the model trained/tested for over 6000 steps, it only reached a precision value of approximately 0.6, indicating that the model is correct at predicting the target or object 60% of the time (at maximum). Thus portraying that the model does not reach high levels of accuracy as the general standard of "great model performance" is recorded to be at an average of 70% precision.

8.2.3 Recorded Recall

Recall data refers to the proportion of actual or true positives being identified correctly by the model. Recall is calculated with the equation: $\frac{TP}{TP+FN}$ where TP and FN respectively refer to True Positives and False Negatives, and is expressed within the following graph/curve:

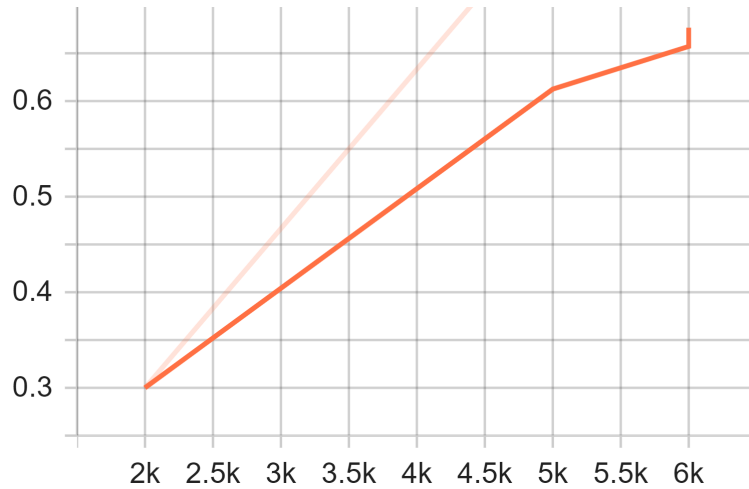


Figure 14: Recall Curve

Figure 14 indicates that the constructed model has a recall of 0.67, stating that it identifies one of the 4 designated targets correctly 67% of the time. Given that an ideal system is designed to approach a recall value of 1, it can be stated that this model is not the most optimal/efficient at object detection.

8.2.4 Comments

This section aims to address the overall viability of the constructed machine learning algorithm. As can be inferred from the previous subsections, despite indicating favourable trends with the loss curves, the system's ability at prediction and detection is comparatively lower than an average standard algorithm. This can be accounted to the number of images used to train and test the algorithm, as a total of 200 images is considered a small data set in the machine learning community. This limitation to the project was due to the need to narrow high quality **synthetic** images available on the internet. Having access to only fake or synthetic images further brings down the viability of the algorithm as it indicates this cannot be utilised in realistic scenarios.

However, the project supports the research conducted and indicates there is potential for machine learning in the field of debris detection and/or identification, provided appropriate baseline resources and materials.

8.3 3D Modelling & Animation

This section of **Discussion** shall be comparatively brief due to most the WOMBAT's hypothetical dimensions being provided within the "Results" section of this thesis report. Among the three types of manual capture scrutinized previously, it was decided that the WOMBAT would hypothetically be equipped with a net/sheet mode of capture. This selection was inline with its previously discussed advantages/disadvantages, specifically being more prominent in capturing space debris < **10cm**.

As shown in Figure 11, the model was equipped with an extension that protrudes past the jaws of the robot. The tip of this extension is then expected to launch the net over the body of the targeted debris to successfully capture it. The debris is then pulled into its storage compartment where the two sliding legs/gates of WOMBAT shall prevent it from escaping. The decision to either recycle or dispose of said debris relies on the design's ability to interact with the ISS on a constant basis.

An investigation of this complexity no doubt has multiple degrees of challenges; with respect to the 3D design of the actual robot, it is hoped that further discovery and propulsion methods will be explored in the future. The model remains hypothetical and if the project were to be continued, more research into professionally designing the WOMBAT would be conducted; this shall be further discussed within this thesis' "Future Work".

8.4 Market & Political Disputes

This phase of the thesis aims to evaluate the various interferences or disputes that could deter the introduction of the WOMBAT or similar robots into the market. Whilst the market shows an astounding demand for this type of product, valuing this field at *USD 1,527.7 million* by 2029 [B (2021)], it is important to discuss how these types of projects interact with the current space treaties governing space activity. Within the list of space treaties imposed, the following four display potential interference [Nations (1966)]:

- “The exploration and use of outer space shall be carried out for the benefit

and in the interests of all countries and shall be the province of all mankind”.

- “States shall be responsible for national space activities whether carried out by governmental or non-governmental entities”.
- “States shall be liable for damage caused by their space objects”.
- “States shall avoid harmful contamination of space and celestial bodies”.

Despite the dawn of the space exploration vowing to benefit "all mankind", modern society has made space activity a race between individual countries' technologies. Currently different countries are working towards being capable of continuously releasing satellites and rockets into space; further building upon space debris and the Kessler syndrome. Another set of information that could interfere with the functioning of robots like the WOMBAT includes being aware of the number of satellites/devices situated in space. In the previously stated moon incident [Dunn (2022)], the property liability behind the colliding space debris was suspected to belong to China; however, no government has stepped forward to lay claim on the event. Multiple governments had and have also installed security and surveillance satellites in space and robots like WOMBAT interacting with such gear may result in a breach of personal information and security. This also indicates that there may be debris or fragments in space from satellites that were not publicly registered/recorded to have been placed in orbit (for potential spy purposes). As such, it is important to note that a plethora of factors such as the above may come into play prior to officially developing such a product.

8.5 Section Overview

This formally concludes the thorough evaluation of the various aims or goals that were set to be achieved as part of this thesis' scope. A further breakdown of the potential in this project for future developments and other avenues of interests that would like to explored shall be mentioned within this report's "Future Work".

9 Conclusions

Having addressed every aim and their respective section within this thesis, a succinct summary of the work conducted shall now be provided. The thesis' initial evaluation of a kernel-based image processing program indicated the weaknesses in its designs: its inability to function in realistic scenarios due to its pixel/color dependency and lack of greater mathematical calculations that could have enabled it to accurately 'bound' space debris.

As such, the thesis shifted to its target to verifying the viability of its machine learning algorithm. Despite fairly accurate loss curves, a max precision of 0.6 and recall of 0.67 hinder its applicability. Another factor that held back the system was the lack of access to real space entity images; burdening the system with its synthetic image dependency.

The secondary goal provided a brief hypothetical representation of the thesis's desired robot, the WOMBAT with net capture, to achieve debris collection. However, further research would be required to finish designing its structure, means of navigation, etc. It was also noted how simply designing and building robot would not be sufficient to introduce it into the market and it would require means to navigate between the current space treaties and/or security concerns.

In a summary, the thesis achieved the various desired goals stated in the initial scope, in the software field. With access to appropriate resources and contacts, the software would be further improved and the hardware/model could be more detailed and professionally designed. This project hopes to generate a sense of urgency towards addressing this thesis' problem statement as the impending threat of space debris exponentially increases over time and human-space advancements.

10 Future Work

This section aims to focus on the next steps or future work that this project would go through, if it were to be continued. Firstly, it would be necessary to contact and gain information and potential real images from existing space organisations. This would allow both the kernel-based program and the machine learning algorithm to function utilising real data rather than synthetic work. The project could then explore other means/sensors for space debris detection such as radar, lidar, etc., and revolve around testing these systems' viability towards debris detection and collection.

Secondly, further details regarding the WOMBAT would be mapped. In-depth research into the type of materials needed to build the external and internal structure such that it can adapt to the changing conditions of space. Other aspects of the design, such as the type of propulsion necessary, are to be integrated into the design for robot to navigate within the Low Earth Orbit. It would also be beneficial to design means to transfer the WOMBAT's payload either into a collapsing orbit or through back and forth delivery between the robot and the International Space Station. With adequate resources and backing, the new WOMBAT model could contemplate cascading multiple capture mechanisms that complement each other, taking inspiration from Lv et al. (2022)'s literature.

11 Bibliography

References

- B, F. B. I. (2021), ‘Space debris monitoring and removal market size’.
URL: <https://www.fortunebusinessinsights.com/space-debris-monitoring-and-removal-market-104070>
- Dunn, M. (2022), ‘Space junk on 5,800-mph collision course with moon’.
URL: <https://apnews.com/article/spacex-science-business-china-moon-7044a26d9b4b43dc44e961c250186f26>
- ESA (2020), ‘The cost of space debris’.
URL: https://www.esa.int/Space_Safety/Space_Debris/The_cost_of_space_debris
- ESA (2021), ‘Space environment statistics · space debris user portal’.
URL: <https://sdup.esoc.esa.int/discosweb/statistics/>
- Gupta, S. (2022), ‘The 7 most common machine learning loss functions’.
URL: <https://builtin.com/machine-learning/common-loss-functions>
- Lv, S., Zhang, H., Zhang, Y., Ning, B. & Qi, R. (2022), ‘Design of an Integrated Platform for Active Debris Removal’, *MDPI* pp. 1–8.
- Matignon & de Gouyon, L. (2019), ‘The Kessler syndrome and space debris’.
URL: <https://www.spacelegalissues.com/space-law-the-kessler-syndrome/>
- Mehrholz, D., Leushacke, L., Flury, W., Jehn, R., Klinkard, H. & Landgraf, M. (2002), ‘Detecting, Tracking and Imaging Space Debris’, pp. 1–7.
- Nations, U. (1966), ‘Treaty on principles governing the activities of states in the exploration and use of outer space, including the moon and other celestial bodies’.
URL: <https://www.unoosa.org/oosa/en/ourwork/spacelaw/treaties/introouterspacetreaty.html>
- Pal, S., Raj, A. B. & R.Shinde, S. (2020), ‘Performance Study of Different Robotic Manipulator Systems Designed for Space Debris Management’, *Fifth International Conference on Communication and Electronics Systems* pp. 1–6.

Parness, A., Abcouwer, N., Fuller, C., Wiltsie, N., Nash, J. & Kennedy, B. (2017), ‘Lemur 3: A Limbed Climbing Robot for Extreme Terrain Mobility in Space’, *IEEE International Conference on Robotics and Automation (ICRA)* pp. 1–7.

Perez, M., Musallam, M., Henaff, P., Garcia, A., Ghorbel, E., Imsaeil, K. & Aouada, D. (2021), ‘Detection Identification of On-orbit Objects using Machine Learning’, *8th European Conference on Space Debris* pp. 1–10.

Rabie, P. (2021), ‘The spacecraft cemetery: Why NASA dumps its trash in the middle of the Pacific Ocean’.

URL: <https://www.inverse.com/science/why-nasa-uses-point-nemo-as-a-graveyard>

Ratner, P. (2018), ‘How the kessler syndrome can end all space exploration and destroy modern life’.

URL: <https://bigthink.com/surprising-science/how-the-kessler-syndrome-can-end-all-space-exploration-and-destroy-modern-life/>

Renotte, N. (2020), ‘Tensorflow object detection walkthrough’.

URL: <https://github.com/nicknochnack/TFODCourse>

Singh, P., Chand, D. S., Pal, S. & Mishra, A. (2020), ‘Study of Current Scenario Removal Methods of Space Debris’, *International Journal of Mechanical and Production Engineering Research and Development* **10**, 1–15.

Srikrishnan, S., Dash, D. P., Pillai, D. S. N. & Arunvinthan, S. (2015), ‘An Approach for Space Debris Clearing using Space Based Robots’, *International Journal of Engineering Research And Management* **2**(6), 1–5.

Suresh, S. (2022), ‘space-debris-detection’.

URL: <https://github.com/simransuresh/space-debris-detection>

SUT (2022), ‘Space junk is dangerous, but it’s also an opportunity’.

URL: <https://www.swinburne.edu.au/news/2022/03/space-junk-is-dangerous-but-its-also-an-opportunity/>

Tensorflow (2017), ‘Tensorflow 2 detection model zoo’.

URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2det

Tingay, S., Kaplan, D. L., McKinley, B., Briggs, F., Wayth, R. B., Hurley-Walker, N., Kennewell, J., Smith, C., K.Zhang, Arcus, W., Bhat, N. D. R., Emrich, D., Herne, D., Kudryavtseva, N., Lynch, M., Ord, S. M., Waterson, M., Barnes, D. G., Bell, M., Gaensler, B. M., Lenc, E., Bernardi, G., Greenhill, L. J., Kasper, J. C., Bowman, J. D., Jacobs, D., Bunton, J. D., deSouza, L., Koenig, R., Pathikulangara, J., Stevens, J., Cappalio, R. J., Corey, B. E., Kincaid, B. B., Kratzenberg, E., Lonsdale, C. J., McWhirter, S. R., Rogers, A. E. E., Salah, J. E., Whitney, A. R., Deshpande, A., Prabu, T., Shankar, N. U., Srivani, K. S., Subrahmanyam, R., Ewall-Wice, A., Feng, L., Goeke, R., Morgan, E., Remillard, R. A., Williams, C. L., Hazelton, B. J., Morales, M. F., Johnston-Hollitt, M., Mitchell, D. A., Procopio, P., Riding, J., Webster, R. L., Wyithe, J. S. B., Oberoi, D., Roshi, A., Sault, R. J. & Williams, A. (2013), ‘On the Detection and Tracking of Space Debris using the Murchison Widefield Array. I. Simulations and Test Observations demonstrate feasibility’, *The Astronomical Journal* **146**(103), 1–9.

Tzotalin (2016), ‘Labelimg’.

URL: <https://github.com/heartexlabs/labelImg>

UWA (2022), ‘Trillion dollar future “space economy” threatened by debris’.

URL: <https://scitechdaily.com/trillion-dollar-future-space-economy-threatened-by-debris/>

Wall, M., Malik, T., Weitering, H., Bartels, M., Gohd, C., Pultarova, T. & Cox, A. (2013), ‘Worst space debris events of all time’.

URL: <https://www.space.com/9708-worst-space-debris-events-time.html>

12 Appendices

12.1 Appendix A: Graveyard Orbit

Figure removed due to copyright restriction

Figure 15: The Graveyard Orbit

12.2 Appendix B: Earth's Orbits and Satellites

Figure removed due to copyright restrictions

Figure 16: Earth's Orbits and Satellites

12.3 Appendix C: Code Listings

```
#Import all the required libraries that enable function
import numpy as np
import cv2 as cv
#Setting the lower and upper boundaries based on known color pixel values.
#Since the program works under the assumption that "all regions but debris
#is black in space", the boundaries range from black to white.
LowerBounds = np.array([0, 0, 0])
UpperBounds = np.array([0, 0, 255])
#Captures the provided .mp4 file for later use
cam = cv2.VideoCapture("debrisV3.mp4")
#Sets up two (open and close) filtering kernels
kernelOpen = np.ones((5, 5))
kernelClose = np.ones((20, 20))
```

Listing 1: Kernel-based Image Processing: Initializations, capturing the video and setting up the kernels

```
while(True):
    #Captures the return boolean value and image frame
    arg, img=cam.read()
    #Resizing the captured frame/video to 1366x768
    img=cv2.resize(img,(1366,768))
    #converts color type from BGR to HSV
    imgverHSV= cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
    # Generates the mask based on the previously set boundaries
    mask=cv2.inRange(imgverHSV,lowerBound,upperBound)
    #Morphologically manipulating the mask
    maskOpen=cv2.morphologyEx(mask,cv2.MORPH_OPEN,kernelOpen)
    maskClose=cv2.morphologyEx(maskOpen,cv2.MORPH_CLOSE,kernelClose)
    #Determines the locations upon which contours will be later placed upon
    conts,h=cv2.findContours(maskClose.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
    #Draw white contours around 'found' regions
    font = cv2.FONT_HERSHEY_SIMPLEX
    cv2.drawContours(img,conts,-1,(255,0,0),3)
    #Draws the bounding rectangles and respective text
    for i in range(len(conts)):
        x,y,w,h=cv2.boundingRect(conts[i])
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
        cv2.putText(img, str(i+1),(x,y+h),font,1,(0,255,255))
    #Opens a window replaying the video with the overlaid text and contours
    cv2.imshow("camera",img)
    #Sets up a key to destroy any open windows upon command
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
    cv2.destroyAllWindows()
```

Listing 2: Kernel-based Image Processing: Resizing, color conversion, mask generation and manipulation, contouring and displaying final output

```

#Install the necessary opencv packages and files
!pip install opencv-python

# Import essential packages for the algorithm
import cv2
import uuid
import os

#An array of to-be-identified/set labels
labels = ['debris', 'astronaut', 'satellite', 'extras']

#Generates a path within the virtual environment folder
IMAGES_PATH = os.path.join('Tensorflow', 'workspace', 'images', 'collectedimages')

#Checks if image path exists, if not re-execute command
if not os.path.exists(IMAGES_PATH):
    !mkdir {IMAGES_PATH}
#Adds four label-based folder to the image path
for label in labels:
    path = os.path.join(IMAGES_PATH, label)
    if not os.path.exists(path):
        !mkdir {path}

```

Listing 3: Machine Learning Algorithm: Initializations and Image Collection

```

#Installing necessary updates for later steps
!pip install --upgrade pyqt5 lxml

#Constructs a new file path within Tensorflow
LABELIMG_PATH = os.path.join('Tensorflow', 'labelimg')

#Checks for new path and clones the required repository
if not os.path.exists(LABELIMG_PATH):
    !mkdir {LABELIMG_PATH}
    !git clone https://github.com/tzutalin/labelImg {LABELIMG_PATH}

#Install the necessary files required for labelling tool from github
!cd {LABELIMG_PATH} && pyrcc5 -o libs/resources.py resources.qrc

#Within this path, opens the labelling tool
!cd {LABELIMG_PATH} && python labelimg.py

```

Listing 4: Machine Learning Algorithm: Image Labelling

```

#import os
import os

#Sets up the pre-trained model
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2
    ↪ /20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'

#Creates the necessary paths for training and testing
paths = {
'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
'SCRIPTS_PATH': os.path.join('Tensorflow', 'scripts'),
'APIMODEL_PATH': os.path.join('Tensorflow', 'models'),
'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace', 'annotations'),
'IMAGE_PATH': os.path.join('Tensorflow', 'workspace', 'images'),
'MODEL_PATH': os.path.join('Tensorflow', 'workspace', 'models'),
'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace', 'pre-trained-
    ↪ models'),
'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace', 'models',
    ↪ CUSTOM_MODEL_NAME),
'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME,
    ↪ 'export'),
'TFJS_PATH': os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME,
    ↪ 'tfjsexport'),
'TFLITE_PATH': os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME,
    ↪ 'tfliteexport'),
'PROTOC_PATH': os.path.join('Tensorflow', 'protoc')
}

#Configuring file dictionaries
files = {
'PIPELINE_CONFIG': os.path.join('Tensorflow', 'workspace', 'models',
    ↪ CUSTOM_MODEL_NAME, 'pipeline.config'),
'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME)
    ↪ ,
'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}

#Adds all file paths into the virtual environment's folder path
for path in paths.values():
    if not os.path.exists(path):
        !mkdir {path}

```

Listing 5: Machine Learning Algorithm: Training & Detection Phase - Part 1

```

#install import wget
!pip install wget
import wget

#clones the required object detection repository
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', '
↳ object_detection')):
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}

#Accesses and install the necessary protocol buffers and object detection API
url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc
↳ -3.15.6-win64.zip"
wget.download(url)
!move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
!cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH
↳ '], 'bin'))
!cd Tensorflow/models/research && protoc object_detection/protos/*.proto --
↳ python_out=. && copy object_detection\\packages\\tf2\\setup.py setup.py &&
↳ python setup.py build && python setup.py install
!cd Tensorflow/models/research/slim && pip install -e .

#Run the verification script
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', '
↳ object_detection', 'builders', 'model_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}

# Missing packages and libraries
!pip install tensorflow --upgrade
!pip uninstall protobuf matplotlib -y
!pip install protobuf matplotlib==3.2
pip install protobuf==3.20.*
pip install scipy
pip install Pillow
pip install pyyaml
import object_detection

# Downloads the SSD Mobnet pre-trained model
wget.download(PRETRAINED_MODEL_URL)
!move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
!cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'
↳ }

```

Listing 6: Machine Learning Algorithm: Training & Detection Phase - Part 2

```

#Construct and correlate the labels and IDs to the map
labels = [{ 'name': 'Debris', 'id':1}, { 'name': 'satellite', 'id':2}, { 'name': '
↳ astronaut', 'id':3}, { 'name': 'extras', 'id':4}]
with open(files[ 'LABELMAP'], 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\t' + label[ 'name'] + '\n')
        f.write('\tid:\t' + label[ 'id'] + '\n')
        f.write('\n')

#install pandas
pip install pandas

#clone the TF record generation file from github
if not os.path.exists(files[ 'TF_RECORD_SCRIPT']):
    !git clone https://github.com/nicknochnack/GenerateTFRecord {paths[ '
↳ SCRIPTS_PATH']}

#Generate the train and test TF records for machine learning
!python {files[ 'TF_RECORD_SCRIPT']} -x {os.path.join(paths[ 'IMAGE_PATH'], 'train')}
↳ -l {files[ 'LABELMAP']} -o {os.path.join(paths[ 'ANNOTATION_PATH'], 'train.
↳ record')}
!python {files[ 'TF_RECORD_SCRIPT']} -x {os.path.join(paths[ 'IMAGE_PATH'], 'test')}
↳ -l {files[ 'LABELMAP']} -o {os.path.join(paths[ 'ANNOTATION_PATH'], 'test.
↳ record')}

```

Listing 7: Machine Learning Algorithm: Training & Detection Phase - Part 3


```

#Copies the configuration files into the training folder
!copy {os.path.join(paths[ 'PRETRAINED_MODEL_PATH' ], PRETRAINED_MODEL_NAME, '
    ↳ pipeline.config')} {os.path.join(paths[ 'CHECKPOINT_PATH' ])}

#import necessary files and packages
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format

#Configuring and Setting up Pipeline Architecture
config = config_util.get_configs_from_pipeline_file(files[ 'PIPELINE_CONFIG' ])
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files[ 'PIPELINE_CONFIG' ], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)

pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths[ '
    ↳ PRETRAINED_MODEL_PATH' ], PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files[ 'LABELMAP' ]
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path
    ↳ .join(paths[ 'ANNOTATION_PATH' ], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files[ 'LABELMAP' ]
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.
    ↳ path.join(paths[ 'ANNOTATION_PATH' ], 'test.record')]
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files[ 'PIPELINE_CONFIG' ], "wb") as f:
    f.write(config_text)

```

Listing 8: Machine Learning Algorithm: Training & Detection Phase - Part 4

```

#Run the training phase
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', '
    ↪ object_detection', 'model_main_tf2.py')
command = "python {} --model_dir={} --pipeline_config_path={} --num_train_steps
    ↪ =6000".format(TRAINING_SCRIPT, paths['CHECKPOINT_PATH'], files['
    ↪ PIPELINE_CONFIG'])
print(command)

# Run the evaluation script
command = "python {} --model_dir={} --pipeline_config_path={} --checkpoint_dir={}
    ↪ .format(TRAINING_SCRIPT, paths['CHECKPOINT_PATH'], files['PIPELINE_CONFIG'],
    ↪ paths['CHECKPOINT_PATH'])
print(command)

# Import and re-import certain files and packages
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util

# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=
    ↪ False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-9')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections

```

Listing 9: Machine Learning Algorithm: Training & Detection Phase - Part 5

```

#Run the testing phase
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
category_index = label_map_util.create_category_index_from_labelmap(files [ '
↳ LABELMAP' ])
IMAGE_PATH = os.path.join(paths[ 'IMAGE_PATH' ], 'test', 'depositphotos_565812646-
↳ stock-video-render-animation-asteroids-field-deep.jpg')

img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
               for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections['detection_boxes'],
    detections['detection_classes']+label_id_offset,
    detections['detection_scores'],
    category_index,
    use_normalized_coordinates=True,
    max_boxes_to_draw=5,
    min_score_thresh=.8,
    agnostic_mode=False)

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()

```

Listing 10: Machine Learning Algorithm: Training & Detection Phase - Part 6