University of Southern Queensland

Faculty of Health, Engineering and Sciences

UAS AUTONOMOUS DROP ZONE ASSESSMENT AND ADAPTIVE DELIVERY

A dissertation submitted by

Mr Christopher Bourke

In fulfilment of the requirements of

Bachelor of Engineering (Honours) (Mechatronics)

October 2022

Abstract

The aim of this dissertation was to research and implement a stereoscopic machine vision sensor system on an uncrewed aerial system. The variables affecting the quality and computational cost of the stereo algorithm were assessed and optimised to be employed on a low-cost companion computer, the Raspberry Pi 4 (4GB). The algorithms used proved computationally expensive and the results tested the limit of the companion computer’s abilities to reach the minimum operating speeds required for UAS logistic delivery missions. The outcomes demonstrated an accurate and dense disparity map on benchmark datasets but the image sizes and tuning in field testing demonstrated a limited efficacy in application.

Measured distance error slightly increased across the effective ranges where disparity was calculated. The cumulative errors from field of view and focal length, taken from the manufacturers specifications rather than directly assessed, and the baseline measurement which all effect the distance measurement after disparity is calculated.

From the implementation and assessment carried out in this paper it is evident that the largest factor affecting the calculation speed of the system is the captured image size and represents the limiting factor in employing this implementation in its current configuration.

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Christopher Bourke

Student Number: [redacted]

Acknowledgements

Dr. Tobias Low

Mr Philip Moss

To my wife. Thank you.

Table of Contents

List of Figures

List of Tables

Nomenclature

Δz - change in depth detected by the stereo setup

z - Calculated distance fom the camera setup

b – Baseline or distance between the cameras in the stereo camera setup

f- focal length of the cameras

Δd - the physical change in distance from the camera setup

Glossary of Terms

API – Application Programming Interface

CASA – Civil Aviation Safety Authority

CNN – Convolutional Neural Network

CSI – Camera Serial Interface

DLT - Direct Linear Transformation

GUI – Graphic User Interface

StereoBM – OpenCV Stereo block matching algorithm

StereoSGBM – OpenCV Stereo semi global block matching algorithm

TFMic – Tensor Flow for Microcontollers

USB – Universal Serial Bus

Chapter 1 - Introduction

“In a properly automated and educated world, then, machines may prove to be the true humanizing influence. It may be that machines will do the work that makes life possible and that human beings will do all the other things that make life pleasant and worthwhile.”

― Isaac Asimov, Robot Visions

1.1 Outline of the study

The development of UAS technology and its integration into society drive each other as convenience, safety and applications expand. Studies and functions that expand the applications of UAS technology especially in the hobby space will drive consumer interest and therefore investment for the future. This study aims to implement a stereo machine-vision based distance measurement solution used to modify a logistic UAS delivery.

1.2 Introduction

Uncrewed aerial systems (UAS) are being employed in multiple sectors and the continuation of development of machine vision and autonomous machine learning means that UAS will be used more broadly and with greater effect in the future.

Logistic delivery has been investigated as an avenue of UAS implementation and is being developed as a commercial solution sporadically since 2013 and, more recently, growth has begun in Australia with two major competitors, Wing and Swoop Aero taking part in CASA trails (Civil Aviation Safety Authority 2022). The employment of these systems is highly regulated in accordance with CASA regulations and while autonomous navigation and control have been the focus of many research papers, efficient delivery has been traded against safety of operations and, critically, the increase in trust of UAS.

The delivery method can be refined by assessing not only UAS position but also the vector required to safely deliver to a selected delivery point. This research seeks to assess the use of implemented stereo vision as a distance measurement method and utilise this method as a primary means of assessing a delivery area while retaining accuracy and maximising efficiency of UAS endurance.

1.3 The Problem

Distance measurement to obstacles and for positioning are required as one of the numerous safeties for UAS operation, particularly in controlled airspace (Civil Aviation Safety Regulation 1998). Many such sensors are active types which transmit a signal to sense distance. These technologies have certain limitations in application and accuracy especially in changing environments often requiring frequency allocation and infrastructure. This study seeks to utilise passive sensors, likely to reduce transmitter noise in the environment, without a delivery marker to expand the flexibility of the delivery system to better operate in dynamic or remote environments with greater options for expansion.

1.4 Research Objectives

The aim of this research is to design and implement a UAS based delivery system which can autonomously assess a designated delivery point for obstruction and transition its delivery method accordingly. The objectives are:

1. Conduct initial research on UAS machine vision applications. Explore UAS delivery methods and the legislation surrounding autonomy in UAS applications.

2. Review methods of calculating depth from stereo vision including the factors most affecting the accuracy and resolution of the output. Investigate differing algorithms for stereo vision.

3. Conceptualise the components of a suitable integrated system including UAS platform, machine vision components, companion computer, ground control, store delivery and testing systems.

4. Select hardware and a suitable software development environment. The requirements for necessary capability and costs to inform selection.

5. Develop a machine vision algorithm for optimized disparity mapping.

6. Construct an initial prototype to facilitate data collection. Including UAS control programs and fail-safes.

7. Refine stereo vision matching algorithms and post process data output for optimum use in a UAS platform.

8. Integrate and deploy the prototype and algorithms at a suitable location and record data for evaluation.

9. Process and evaluate experimental data.

Employment on a UAS platform dictates a low power companion computer and camera system to maintain typical mission planning and execution in addition to delivery point assessments.

Depth resolution of 2m at 40m slant range to the delivery point forms the basis of a realistic operational envelope, avoiding obstructions, i.e., trees and buildings while maximising UAS flight times. The design will maximise this sensing distance while retaining the depth resolution.

The delivery method decision times and depth calculations must be valid for operation at a frequency of at least 3 per second. These decision times will inform maximum approach speeds 5 -15 m/s is an acceptable range.

1.5 Conclusion

The aims of this research are to define the factors that impact the implementation of effective autonomous UAS delivery and move to design and test a system that satisfies the research base and demonstrates the required technical engineering ability. The expected outcome of this implementation should yield tangible results supporting the stereo-vision hypothesis discussed below while also highlighting the practicalities of any implementation of autonomy or machine vision. The literature review in Chapter 2 will investigate the current state of play of UAS in the logistics area, the typical sensors required for sensing the environment, the background and applications of machine and stereo vision, including the hardware that may be required.

Chapter 2 - Literature Review

2.1 Introduction

Research has been conducted into computer vision systems since the late 1960s (Alam 2020). There has been a continual refinement of this process branching out into other emerging technology streams, such as machine learning, with numerous vision systems employed throughout all areas of technology. This literature review will explore the current state of UAS in logistic chains and how the technology is shaping this area, the infrastructure and policy in place to support the growth of UAS and also the commercial trends in the investment and deployment of these technologies will also be reviewed.

The typical sensors combined with UAS and how they are utilised to produce safe and effective deliveries including mandated controls and in differing deployment environments (agriculture, logistics, hobby, etc.). This will focus on the flexibility and adaptability of technologies and particularly the use of stereo machine-vision.

Computer vision and stereo vision will be assessed in deciding a direction for the improvement and implementation of a UAS model and the factors that would affect its capability. Factors that affect the design of the model will be reliant on commonly accessible "hobby grade" equipment as this will be the driver for future development and employment.

2.2 UAS in logistics

2.2.1 The last mile problem

The "last mile" issue facing logistic chains is a problem that arises due to the complexity of the final portion of the delivery of an item to the consumer. Whereas manufacturers focus their attention on the distribution of their products in the design of warehouses and despatch facilities and global logistics are similarly robust and fit for purpose, the complexities of optimum routing, speed and accuracy of delivery, package security and overall cost are far more difficult to assess and control in the "last mile"(Ranieri 2018)(Diaz 2021).

Ranieri (2018) cites 28% of total cost attributed to last mile delivery while Diaz (2021) cites up to 53% of total shipping costs. Increases in efficiency in this area would obviously lead to substantial increase in profits for logistics companies. It is likely, therefore that investment in this area would be prioritised.

Populations world-wide are increasing but not expanding, becoming more dense (Macrotrends 2022) leading to logistics chains requiring more refined last mile solutions. This could mean larger vehicles or more frequent delivery trips. The most energy efficient solution could be the expanded commercial use of logistics UAS in high density urban environments. The COVID-19 pandemic served as a warning for population density (Bowditch, 2020), however, and may curb this trend altering the direction of urban densities. Investment trends of the Australian Government appear to reinforce population density to maximise the infrastructure investment. This obviously makes disease vectors more effective and may change this approach to population density which could have effect on the use cases for UAS.

Muller, Janke and Rudolph (2018) make the distinction from pure technical solutions to UAS investment to state that any improvements to automation or transitioning to electric transport would not be an effective long-term solution rather suggesting "policy-supported reduction of disruption of efficiency limitations." This seems like a forward-thinking approach that has merit given the levels of technical ability for UAS delivery and automation is currently at a high standard and used in a wide variety of areas, it is more likely the adoption of this technology from a governance standpoint, whereby aging laws controlling operations from 1998 (Civil Aviation Safety Authority) do not take into account the future deployment of these technologies on a larger scale.

2.2.2 Logistic UAS civilian and military

European Cockpit Association (2020) posits that the uptake of UAS in the near future is not feasible and even whether their use would be wanted. Further, the integration of fully autonomous systems into spaces where humans operate raises issues regarding "…fundamental philosophical, legal, safety, security, and societal issues". The paper relies on the goal of automation being a force multiplier for human operation and not a standalone activity. There is a myriad of automated operations now across every field of human endeavour and the stated necessity for a human to remain in command could be viewed as a function of trust in the UAS not one guided by academic UAS research. Interestingly Pani (2020), shows that 61% of respondents responded positively to the increase in cost required by automated delivery services, when surveyed around the period of the COVID-19 pandemic.

Drone systems have been developed as an enhancement to logistics chains in military and commercial factory settings using ground-based rovers and airborne systems (Ergene 2016) (Rana, Praharaj, & Nanda 2016) (Levin 2016) (Statt 2017). Recently the Civil Aviation Safety Authority has approved the use of UAS for small scale commercial delivery, Wing Aviation Pty Ltd for small scale fast-food related deliveries and Swoop Aero for medical supplies and related deliveries (Civil Aviation Safety Authority 2021). The ranges are 10km for the Wing drone and up to 60km for Swoop Aero, both UAS have differing delivery profiles that likely informs the ranges as appropriate. As developments continue in the fields on drone sensors, navigation, machine vision and necessary hardware the use of UAS will become widespread.

Shao (2020) cites the viability of UAS in the logistics chain and presents an analysis of the "expected level of safety (ESL) and the factors that contribute to this figure in both the air and ground domain. Two cases studies are reviewed which does not form an exhaustive list of scenarios but forms a robust basis for future assessments of safety and risk.

Autonomous deliveries are attempting to refine the last mile problem with autonomous UAS increasing efficiency and manoeuvrability in dense urban settings while also reducing CO_2 emissions (Rodrigues et al 2022). Bicycle and pedestrian courier deliveries are filling this gap currently and the automation can increase speed of deliveries for small items.

The research demonstrates that last mile problem is affected by multiple factors far outside of any single company's control. Population density, infrastructure, consumer trends, legislation around delivery requirements, vehicle standardisation, real estate availability, ecommerce trends, etc.

In military applications, where the appetite for risk and the requirement for accurate and direct deployment in insecure and volatile environments is paramount, the use of UAS has diversified from surveillance to logistics. With 90% of deliveries being less than 50lb(23kg) (Defense Brief Edittorial 2020) medium scale UAS are being investigated as serious alternatives to large scale logistics assets. In 2020 testing from Naval Air Warfare Center Aircraft Division tested commercially developed UAS with the criteria to move 20lb of cargo 25 miles, 65 platforms were assessed and only 2 managed to partially meet the criteria demonstrating the complex nature of practical implementation from a relatively simple design brief.

The Australian Department of Defence's Strategic Update (2020) and the Defence White Paper(2016), that describes the strategic direction of Australia's defence including acquisition and long term policy, describes limited forecast acquisition of UAS focussing solely on tactical combat systems rather than logistics. This direction could be a result of the extremely large distance required to cross the country and to get to the areas of interest across the world that are currently serviced by the traditional platforms available in defence and given coalition agreements the utilisation of other nation's assets in country may be more viable.

2.2.3 Increased investment

The state of infrastructure Australia is not currently setup for established logistics UAS. The Australian Government (2021) has published the National Emerging Aviation Technologies policy which describes the intended increase in utilisation of UAS with policy and infrastructure support required to safely introduce and sustain long term logistics chains. The policy cites "safer, cheaper and more efficient movement of people and goods" with intentions of targeting regional and remote areas which represent 28% of the Australian population (Australian Institute of Health and Welfare 2022). The policy also describes the practical requirements of RF allocation and management with typical operation vastly increasing the requirements of civilian frequency availability and security of RF radiation, also suggesting dedicated UAS corridors to refine traffic control and increase safety in dense populations. The Government's Emerging Aviation Technology Partnership (2021) is a $32.6 million allocation over two years aimed to address community needs particularly in regional Australia. The Victorian Government's Advanced Air Mobility Industry Vision Statement (2022) reflects the Federal Government's direction including regulatory innovation prioritising agility and action proportionate to risk, this is a departure from the current state of policy control dated 1998, notably climate action is a primary driver for this initiative which supports the research found in the literature.

2.3 Delivery profiles and efficiency

UAS delivery methods gained popularity in 2013 with Wing, Zipline and Skypro with Amazon's Prime Air also getting investments from large logistics chains such as DHL, FedEx and UPS in America. These drones' delivery methods have not developed as quickly as expected especially the last mile taking up 40% of delivery costs (Wendover productions 2021) a large part of this is the difficulty to implement advancements creating cohesive environment for operations with air traffic control systems such as CASA and FAA.

Early delivery solutions relied on landing the UAS in a relatively uncontrolled area which required a high level of autonomy to ensure safety. Newer designs use a hover and winch approach for private residences (Wing 2021) or drop a parachute fitted payload from height(Swoop Aero 2022). The difference in ranges of these two approaches while having similar UAS platforms speaks to the efficiency of the delivery methods.

Australia currently has two commercial logistics UAS operators trialling with CASA approval, Wing and Swoop Aero (Civil Aviation Safety Authority 2021). These companies deliver small payloads from medical equipment and supplies to fast food like Uber Eats with delivery ranges varying from 10 – 60km(Wing 2021)(Swoop Aero 2022).

The Wing UAS utilise a hover delivery method with a winch cable. Transporting small payloads in cardboard containers to personal addresses. Swoop Aero use a VTOL delivery to a "nest" focussing on a ArUco marker style landing zone indicator. The biggest issue surrounding the utilisation of these drones for delivery is the deconfliction with other air traffic and the safety surrounding beyond line-of-sight operation, especially in the delivery stages (Civil Aviation Safety Authority 2021). The factors that most affect the community is noise from the UAS during the delivery which the Department of Infrastructure, Transport, Regional Development and Communications manages in accordance with the Air Navigation (Aircraft Noise) Regulations 2018(Department of Infrastructure, Transport, Regional Development and Communications 2022). There is also an inherent risk of being under flight paths as delivery vectors change to service any number of people in the area that they are approved to operate in or pointedly, the risk of UAS collision with varying obstructions that are undetected in the real-world such a existing infrastructure.

Importantly, the approval of these operations is predicated on the minimisation of noise pollution due to the relatively low-level operation of the delivery drones. Specifically, from the Aircraft Noise regulations 2018, the approval for operations can be revoked by the Secretary if a remotely piloted aircraft "…is likely to have a significant noise impact on the public." Reduction of operational time over the public particularly in the delivery phase would reduce the risk of noise pollution.

Zhang et al (2021) investigated the energy consumption of differing UAS profiles during delivery and flight. The research indicates that delivery operations yield largely fluctuating energy usages although the models reviewed in the paper do not accurately represent the actual energy usage. It is more simple and likely more efficient to utilise a cruise speed, fixed altitude delivery method using a ballistic model for payload delivery. This paper does present several models estimating the energy required for steady drone flight per unit distance (Epm) and as a conservative model indicates that an increase in speed and a reduction in maneuverers results in the furthest range and lowest Epm. Rodrigues (2022) cites a ~30% energy saving when removing vertical take-offs from the UAS path. Robust path planning (Debnath, Omar & Latip 2019) (Bouzid, Bestaoui & Siguerdidjane 2017) (Galvez, Dadios & Bandala 2014) and numerous other methods present autonomous path planning in a 3D space, referenced from the travelling saleman and working into machine learning supported path optimisation.

2.4 Sensor suites used in UAS

Distance detection is a primary navigation tool used in almost all autonomous applications. A majority of the technology available are active sensors that transmit a signal in order to detect distance. This paper seeks to implement a passive sensor in stereo machine-vision and to assess its performance and flexibility.

2.4.1 Machine Vision

The pinhole camera model is a simplification of the complex interaction between the outside world and the lenses or internals of the camera system and how it affects the light striking the camera sensor.

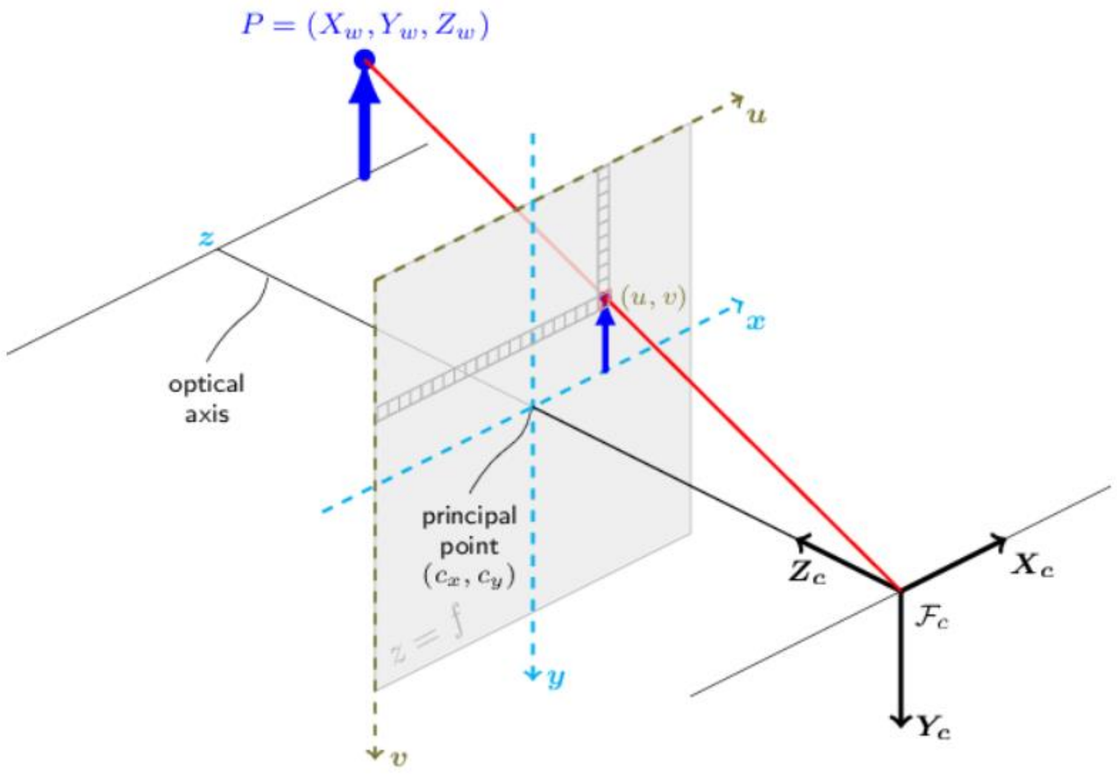


Figure 1 - Pinhole Camera Model Detailing a Real-World point, P, and the point projected on the image Plane at (u,v).(Opencv 2022)

The matrix A represents the essential matrix that effects the real-world points made up of the focal length in the 2D plane, fx and fy, and the principal point of the image plane, cx and cy.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The representation of the point in the image plane [u v 1]^T is the position in the image plane scaled by the factor, s. the essential matrix modifies the world coordinates on the real-world point, P, represented by the real-world coordinates X_c Y_c Z_c

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

The distortion is modelled using the methods described by Louhichi, Fournel, Lavest and Aissia (2007) (OpenCV 2022)and can be solved to yield the distortion coefficients as

$$(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6[, s_1, s_2, s_3, s_4[, \tau_x, \tau_y]]]])$$

2.4.2 Stereo vision

Stereo vision is a passive sensor that uses machine vision and two cameras to calculate depth disparity in viewed image like many biological vision systems. This method relies on feature matching and known camera factors, intrinsic and extrinsic, to create disparity maps, refer to Figure 2. The limitation of the stereo disparity falls inversely with distance from the camera and like most machine vision applications low light operations and low texture environments fail feature matching and therefore are not effectives (Zou & Li 2010).

Stereo vision relies on two, or more, views of an object and the estimation of depth occurs because of the comparative placement of the viewed object in each image plane. Figure 2 references the general setup that enables a disparity between the left and right images planes. There are a number of important features that inform the disparity including the objects distance from the cameras, resolution of the cameras, baseline between the cameras, in addition to intrinsic camera values like focal length and lens corrections.

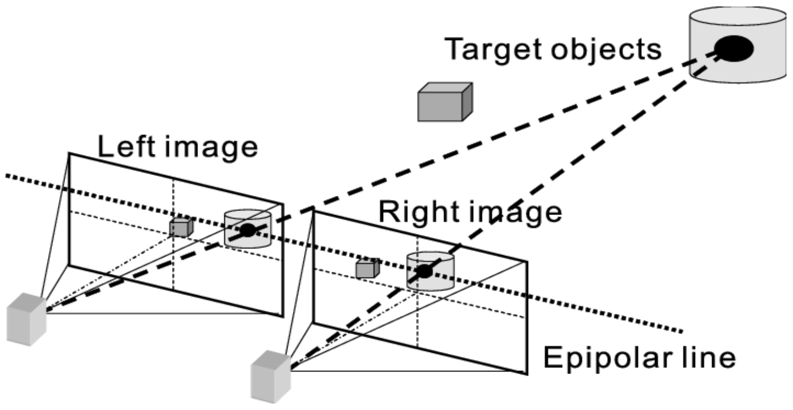


Figure 2 - Stereo Vision Concept showing image planes and disparity of viewed object in space (Hariyama, Masanori & Kameyama 2008)

In order to utilise the images, the most important factor is the ability to match each object in the captured images. There are a number of feature matching techniques available in computer vision applications and a number of variables affecting the capture of images; extrinsic (world to camera): focal length, field of view, lens intrinsic (camera to image and image to pixel representation): field of view, aperture (as it pertains to the pinhole camera model), resolution (OpenCV 2022).

The Middlebury Computer Vision Pages contain multiple stereo datasets developed from the research of D. Scharstein, R. Szeliski, and R. Zabih (2001) and updated periodically. The datasets contain rectified pairs of varying scenes with the details of intrinsic and extrinsic and stereo depth variables. The 2021 dataset has multiple pairs that will form a basis for assessment for the stereo algorithm (D. Scharstein et al, 2014). The computer vision page also presents the research papers, algorithm performance evaluation, and running environment for each algorithm. It is noted that a majority of top performing algorithms utilise high performance GPUs which, as detailed in the hardware assessment, will likely be unobtainable given the aim to deploy this model on a low-cost companion computer.

Stereo vision implementations that have utilised a lower power computer often capture low resolution image pairs with many papers capturing 320 x 240 pixel images for stereo disparity assessment (konlige, 1998)(Cooper et al., 2017). Other papers 640x480 images with higher power computers. There have been successful implementations of low power stereo vision systems Gehrig, Eberli, Meyer (2009) which report some extremely fast frame rates while implementing stereo vision which are well over the assumed requirements for successful UAS deployment.

Research (Sun 1997) (Kumar & Desai 1994) (Chang & Chen 2018) demonstrates the application of image pyramids to increase the speed of the stereo matching process although Chan and Chen combine the pyramidal approach with a 3D Convolutional Neural Network (CNN). The advantage of the pyramidal approach is that broad detail and context is preserved at lower resolutions but also presents vastly less search area to minimise cost.

Camera Calibration

Camera calibration is a critical first stage for any machine vision application. Calibration of the cameras in the system seek to estimate the variables involved in transferring information from the external world to the image plane of the camera. These variables are noted as intrinsic and extrinsic parameters and deal with the effects of the physical sensors and lenses contained in the system. Intrinsic factors represent the camera internal characteristics such as focal length, sensor size, image centre and any distortion, radial or tangential, applied as a function of internal lens placement. The extrinsic parameters relate to the position of the camera system in the outside world and its orientation. With these parameters estimated after the calibration stage the elements of the 2D image can be accurately transposed to the 3D world forming the basis of machine vision-based sensing of the real world. Zhang (2000) presents the widely accepted method for camera calibration in their paper whereby multiple images (at least two) of a calibration target are procured and the image is referenced against the known targets configuration so these parameters can be estimated and applied to all subsequent images captured by the system. A chessboard or ArUco marker are the default selections that can be used as a calibration target as these provide high contrast and sharp edges or corners that are clear and simply identified in machine vision algorithms (OpenCV 2022). Remondino & Fraser (2006) present a number of approaches to camera calibration. The capture of a known calibration target is demonstrated in a number of approaches as in (Heikkilä & Silven 1997) (Tsai 1987) (Zhang 2000) also utilising the pinhole camera model to simplify the extrinsic and intrinsic variables which are used to calculate the essential matrix. Zhang (2000) presents a method for image camera calibration based around the pinhole camera model which has been adopted as the standard for future research into fast and accurate calibration algorithms. OpenCV utilises the pinhole camera model and leverages heavily from the implementation suggested by Zhang.

A direct linear transformation (DLT) is one of the methods used for estimating intrinsic variables Qi, Li, & Zhenzhong (2010) summarises the typical approaches and suggests an artificial neural network to further calculate the cameras radial and tangential distortion for a more realistic non-linear model rather than the simplified pinhole camera model.

Critically, many web cameras are automatically implemented with auto focus functionality enabled and when a calibration target is being moved for capture the camera system can often refocus to best display the target. This function alters the intrinsic parameters of the camera in between calibration target capture and therefore causes the estimation of these assumed static variables to be incorrect (Ricolfe-Viala & Esparza 2021). It is therefore imperative that the autofocus is disabled when camera calibration and operation is underway. While studies have suggested ways in which auto focus can be implemented during the calibration and operation stages the simplicity in fixing the camera focus particularly where fine detail is not required should mean a reduction in algorithm complexity and computational cost.

Rectification

Image rectification is the correction of subsequent images using the estimations of the extrinsic and intrinsic parameters found in the calibration operation. This step is required to accurately represent the captured images as a representation of the outside world critical for machine vision applications but moreso for stereo applications where each system is referenced to its partner.

The stereo rectification process seeks to not only correct the distortion in the images but also to align the captured image pairs to their partner along epipolar lines and reprojecting the image pairs on a common plane. This reprojection along a common plane makes stereo matching and disparity calculation simpler (Pollefeys,Koch & Van Gool 1999).

Epipolar lines

The epipolar plane is defined between the epipolar lines from each camera and the line between the camera's projection centres. The result of the creation of epipolar lines creates a method of aligning observed images between two misaligned camera systems. Effectively observed images exist along the same epipolar line in each image plane. This is the basis of simple stereo matching algorithms which track along the defined epipolar line with some criteria to match the blocks for stereo assessment.

Stereo typical search functions

The research conducted into stereo matching algorithms has some almost universal reference points that many of the current implementations refer to in their development. Birchfield and Tomasi (1999) discusses the calculation of a dense disparity map quickly while dealing with untextured regions and post-processing for complete and neat edges of images. Hirschmuller's (2005)(2007) research into developing the semi global stereo block matching algorithm that forms the basis of OpenCV's disparity map calculations (OpenCV 2022)and Konolige's (2010) StereoBM function.

2.4.3Monocular depth

Carrio et al (2020) presents a method of stereo vision implementation with a machine learning model which identifies a UAS and trains a model from the resulting high resolution (1920 x 1080 pixel) disparity and the depth when viewed from the camera system. Similar approaches are demonstrated in other works which yield accurate results on low power computers (Thorat, 2019). The depth calculations are used to train an early stopping strategy from a 470-image dataset prior to data augmentation to increase the training data and validation sets. This approach utilises a far more powerful computer to capture high resolution images and train the model. The object detection reduces the requirement for dense disparity maps instead highlighting an target and applying stereo matching and distance calculation to it solely. This reduces the image processing cost and leaves overhead for other functions. This application can be implemented on a low power computer to interpret the model particularly the application in Tensor Flow for Microcontrollers (TFmic). Importantly, this model uses a Nvidia Jetson TX2 a device designed for AI execution with considerably more power, 2.5 times more than the Nvidia Jetson Nano, one of the prospective companion computers investigated, as reported by Nvidia (2022). The camera system is a ZED pre-calibrated camera system with a narrow 12cm baseline which autonomously filters disparities to yield continuous disparity maps. The system has a default range from 0.4 – 25m and a maximum range of 40m. With this setup effective training of the machine learning model could be trained for distances up to the experimental goal of 40m but the prohibitive cost of $449 excludes this camera system from selection. There is a similar option in the Intel Real Sense ($499) which has been excluded for the same reasons and a recommended range of up to 6m (Intel Corporation, 2022).

Cherry (2021) demonstrates a reliance on scaling and definition, relative size, absolute and familiar size, elevation, texture gradient, motion parallax, aerial perspective, linear perspective, overlap, shading and lighting. In the employment of a CNN in this study the weights attributed to these variables were not investigated. The work by Saxena, Schulte and Ng (2007), demonstrate detailed depth maps using monocular and stereo cues using Laws' masks and oriented edge filters to assess texture variations. They cite the requirement in biological systems that learnt cues ie. the colour of the sky or grass, the significance of occlusion, blurring and haze that develop as cues to depth and requires the overall context if the image is to be effective. The detail of the stereo correspondence is minimal, but it is defined the sum or absolute difference (SAD) minimising which finds 0.2px disparity which is very accurate. This method is used in other matching criteria in the literature and seems a robust, simple and efficient approach.

Miangoleh et al (2021) presented an algorithm that uses varying resolutions in an image pyramid to develop areas of contextual information that is then blended with higher resolution information that may sacrifice these contextual clues by utilising a smaller field of view. This develops the approach from the MiDAS approach which attempts to optimise the scene structure and high frequency suggesting their approach detail depth discontinuity disagreement ratio.

The trade-off is an intelligent complimentary filter style combination of data that results in high quality depth estimates that exceed MiDAS trained algorithms. The problem is that this algorithm erroneously estimate depth where no physical depth exists eg. Images of a scene or possibly a reflection. This may mean that this type of algorithm may give false indications in dense urban environments where many points of reflection exist.

Figure 3 - Monocular Vision Error with Inferred Depth

The MiDAS algorithms might be difficult to implement on a Rpi but this is the basis for likely the future of depth calculation there are multiple other indication of effective use of machine learning aiding stereo algorithms.

Poggi et al (2008) developed an algorithm that could effectively run at 1.7s per frame on the Rpi 3 from a web camera. Other frequency 2Hz and 150MB of memory. Again, using a pyramidal feature detection to cover the requirement for local and general features that imply depth. Importantly using 720p web cameras have enough disparity and via a robust training network which is enough to output a very accurate depth map.

The report is transparent with its resource allocation and cost and is likely the time for calculation would only be improved with more up to date architectures such as the Rpi 4. However, the drawbacks due to networks training costs and the resources required preclude it from selection as a part of this project. Further work is expected in this field and will likely form the basis of future applications.

The implementation may be improved using machine learning models trained externally and transitioned to more economical routines as with TensorFlow Lite for Microcontrollers (TFmic). This implementation can convert larger models trained through Tensor Flow into Tensor Flow Lite and then into an even smaller version in the TFMic versions to be run on the Cortex-M series, for microcontrollers. The Raspberry pi 4 has a more powerful Cortex-A72 Processor designed for a higher performance standard but also runs an operating system which could bog down the processing speed (Sirin Software 2018).

2.4.4 Optical flow

Optical flow is another monocular distance measurement option. The basics of the optics and structure from movement, there are several methods available to recreate structure from motion that relies on tracking a point from one frame to another across a time step.

Lucas-Kanade(LK) method relies on small movements within a neighbourhood of pixel to track a pixel along a path. With this movement the speeds of the viewing camera and the viewed object can yield structure with which to reconstruct the image and infer depth of the viewed object. Javidnia and Corcoran (2017) state a criterion for accurate LK optical flow calculations. Small motion steps, texture rich features and smooth movements. The method uses ORB feature matching to gather correspondence and utilises the Huber loss function for structure optimisation which effectively reduces the effect of outliers of estimates.

The optical flow method would likely suffer for distance estimation, given that a larger distance from the camera will result in a smaller movement in the image plane. The cost in implementation would be reduced by relying on a region of interest, that is the delivery point, but would increase the likelihood of inaccuracies because of the “barber pole” effect of reduced contextual information regarding flow in the scene.

The equation to infer depth from movement relies on robust tracking of a point between two 2D image frames and data relating to the velocity of the viewed object, velocity of the camera observing the scene, the focal length of the camera and the position of the point in the image relayed in Baraldi, Micheli and Uras’ work (1989). This paper uses newspaper sheets to enhance the texture or optical interest in the scene to better track individual points. This indicates that in an outdoor environment this method may work well when viewing a ground level delivery point. The experiment also displaces the optical and translation axes to reduce noise in the calculated depth as a function of the focus of expansion. The focus of expansion being the point where closing flow vectors meet (O’Donovan 2005).

Equations have been reiterated in several papers, Refer below

$$v_x = \frac{fV_x - V_z x}{Z} \qquad v_y = \frac{fV_y - V_z y}{Z}$$

Figure 4 - Velocity of image point from optical flow (NYU 2012)

This equation presents the velocity of the points in the image plane and rely on focal length, object velocity in the world and the translation velocity along the Z dimension along the path of travel, and the physical depth of the object in the path.

To infer depth, the image plane velocity (vx/vy) can be estimated given two consecutive images with a small time between them to ensure that the tracked points are within a small neighbourhood. This change in position in the image and the change in time (dt) can then be used to estimate image plane velocity and with a known translation of both object and camera the depth can be calculated, which obviously relies on the same point being identified correctly in each image relating to the stereo vision disparity measurement. The camera translation speed Vz will be the velocity of the UAS, the delivery point will be stationery which will reduce the complexity of the calculations.

To enable image matching between the two images colours of the point must be the same. The translation of the viewing camera can not only create occlusion of the image point or in a dynamic world shadow or light may change because of the cameras angle inducing a reflection or change in the surface in the world. HSV colour schemes are less likely to be affected than RGB colour schemes (Mamdouh 2020).

The implementation of the L-K sparse optical flow occurs using pyramidal iterations, looking at high level disparities then refining the resolution to reinforce found movement and introduce finer details. Sparse optical flow can result in robust tracking (NYU 2012) and is useful when larger movements are expected between image capture.

2.5 Hardware

Of note is the relatively high-end hardware that a majority of the most accurate and efficient algorithms use when compared on the Middlebury Vision (Scharstein, D, Szeliski, R, Hirschmüller, H 2022). Li et al (2020) UAS mostly reliant on cloud computing and mobile edge computing (MEC) might be a viable solution to alleviate the latency on the network. This would have flow on effects for infrastructure investment and the wider employment of UAS for monitoring and offboard computing. This may not be possible in regional and remote areas of Australia where remote network infrastructure is unlikely to be appropriate for MEC or offboard computing applications. The placement of markers, common in the popular UAS delivery guidance models may also not be suitable in remote areas in emergency situations or where access is difficult, or personnel are not available. Therefore, a delivery method that does not use a marker would have a large amount of flexibility if the method can be accurately implemented.

2.5.1 Companion computer

The companion computer will be required to calculate the stereo disparity maps and also make decisions via a state machine to pass updates to the mission on the autopilot. The listed examples of the companion computers from the Ardupilot documentation shows numerous ARM-based single board computers with the ability to communicate via MAVLink protocols. There are numerous options that could fit this broad brief, the other influencing criteria are cost, CPU and GPU capacity, camera inputs, power requirement, I/O interface and availability.

The initial shortlist was as below:

- Jetson Nano Development Kit
    - GPU (stereo projects) and GPIO pins
- Raspberry pi compute io 4
    - Can’t fit an external GPU to it
- Raspberry pi 4
    - Has a little GPU might be a problem

The stereo algorithms investigated leveraged heavily on GPU operation given the nature of the matrix operations and so a more sophisticated computational capacity is preferable. The Jetson Nano presents the most powerful GPU of the list and carries a duplicated GPIO set as the Raspberry Pi presenting itself as a preference for selection.

2.5.2 Camera system

There are a number of self-contained stereo camera systems that are pre-calibrated such as the ZED series and the Intel Real Sense. These units demonstrate excellent stereo depth with convenient API control for data collection but are expensive and often have reduced distance given their narrow baseline as detailed in the literature.

There are a number of camera systems available predominately transferring data via USB or CSI. The camera systems should be compatible with the data transfer connections of the companion computer.

The CSI ports are capable of higher data transfer rates than USB 3.0 (Kumar 2022) and are available on many of the low power typical companion computers utilised with UAS.

The cameras available from Raspberry pi are low cost and reliable and come in a variety of resolutions. Given the assessment of an increase in captured image resolution the highest resolution camera available is from the Raspberry Pi HQ camera.

Capturing synchronous images has been demonstrated to be imperative when stereo matching (Zhang 2010) and hardware synchronisation is not always reliable and can introduce lags in real-time operation. Camera multiplexers have been used successfully in some of the studies in the literature and should be a focus for procurement if required.

The camera system requires mounting and control via the UAS meaning a gimbal or mount must be procured. In this case given the flexibility of the final camera system a customised gimbal system could be appropriate given the complexity that multi-DOF gimbals would introduce into the system.

2.5.3 UAS Platform

The platform requires a minimum payload capacity of approximately 1kg (Placek, 2022) to enable it to lift the delivery payload as well as the companion computer, avionics, and the stereo camera rig. A generic commercially available frame would be sufficient.

2.6 Conclusions

For this design the approach is a simple direct point approach at a fixed speed and altitude. The height will be set to avoid ground level obstruction and minimise the effects of wind, approximately 40m, for the least climb and the simplest approach to a point maintaining a fixed altitude. The delivery method will be a cruise speed level altitude release in the case where there is no obstruction. Distance from the target to release the payload will be calculated by a simple 2D continually calculated release point algorithm. In the case of obstruction, the delivery method will be modified to a hover over the delivery point and drop, while not the most efficient, it should result in an accurate delivery.

Chapter 3 - Research Design and Methodology

3.1 Hardware

The basis of the design is to utilise common hobby equipment and implement the stereo distance measurement. In that respect the following selections have been made for investigation:

Companion Computer – Raspberry Pi 4, typical, low cost, companion computer utilised for a diverse range of applications.

Raspberry Pi 4 Model B 4GB RAM (PiAustralia 2022)

- 1.5GHz Quad Core Cortex A72 ARMv8
- 4GB of LPDDR4 SDRM
- Integrated Graphics(OpenGL ES)
- Two USB 3 ports
- Two USB 2 ports
- Two microHDMI ports
- WiFi, Bluetooth and Gigabit Ethernet
- One CSI port

Arducam stereo Hat – This module fits onto the GPIO and allows for twin synchronous image capture and transfer via CSI ports also an external power supply.

Raspberry Pi HQ cam (SONY IMX477) – The HQ camera modules use a CSI connection and are capable of 12.3-megapixel, 4056 x 3040 pixel images with a 1/2.3” sensor format.

Lenses – LN050 16mm focal length lenses with 24°(horizontal) field of view (FOV)

Stereo Gimbal – A 3D designed and printed integrating a SG90 servo and a rigid 1-DOF positioning the cameras in a fixed plane.

UAS frame – S500 quad copter frame with Cube Orange autopilot

Release mechanisms – Eflite EFLA405 servoless payload release modules (4)

Given the global climate of logistics and chip shortages at the beginning of 2022 there was considerable difficulty in purchasing any of the companion computers required for this implementation. A Raspberry Pi 4 B (4GB) was procured with which to begin and although this was a tertiary choice. The cost of the build is detailed in Table 1 below.

Table 2 - Stereo Machine-Vision Implementation Cost

Item	Quantity	Cost
Raspberry pi 4 B (4GB)	1	$92.40
Arducam Stereo Camera Hat	1	-
Raspberry Pi HQ Camera (Sony IMX477	2	-
LN050 16mm Lens	2	$224.99 (kit cost)
CSI ribbon cables	2	$3
SG90 servo	1	$2
E flite servoless payload release	4	$58.16
Total	-	$380.55

3.2 Methodology

The stereo distance measurement process follows the typical methods detailed in the literature following the work detailed in OpenCV's library (2022). Initially it is vital that the cameras are calibrated, and the camera images rectified prior to operation. After the calibration and rectification process is completed, the operational images can be captured and corrected using the calculated extrinsic and intrinsic variables and the stereo matching algorithms can create a depth map. Using the depth map created by the matching algorithm the depth can then be calculated as a function of the focal length, field of view and baseline of the camera system. The following details the stages of the process required to calculate depth from stereo images.

Calibration

The first step of the machine vision accrual of images in the understanding of the intrinsic and extrinsic factors that affect the captured images. There are several factors that arise from the lens and camera sensor manufacture known as intrinsic factors.

Generate the reprojection error which is the pixel difference between the 2D vectors as they would exist in the 3d plane

Undistort

Use the calibration data and apply it to the new images thereby accurately displaying the images as they should appear in the world.

Rectify

The rectification method is critical in the ability to match the windows and create an accurate disparity map. This step ensures that both image planes are aligned horizontally so that matching algorithms can search along a single line of pixels rather than use a larger window or a more global search pattern.

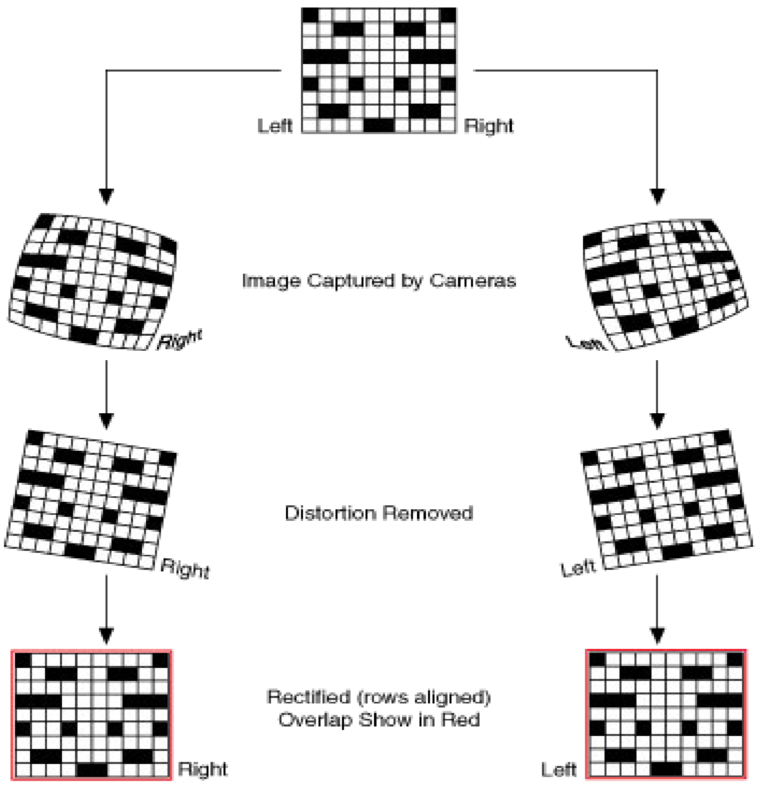


Figure 5 - Stereo Calibration and Rectification procedure (Ni Vision Concepts 2022)

Disparity map

The stereo disparity maps created in this implementation will be made using the functions available in OpenCV's library and follow the methods of Konolige (2010). Using the OpenCV stereo matching algorithms StereoBM the pixel row is checked for matches between the left and right image and if a match is found the difference in the relative positions in the image plane is the disparity for estimating depth.

StereoBM algorithm is a customised block matching approach implementaed by Konolige(2010) an epipolar search matching blocks and minimising the error using SAD, SSD or NCC. The semiglobal block is adapted from Hirschmuller's (2007) work and extends the search pattern to eight directions reducing reliance on accurate rectification and yielding better results but being more expensive.

The stereo disparity calculation algorithm StereoBM uses a block matching algorithm where a defined window tries to match a segment of the left and right images along epipolar lines of the rectified images.

The algorithm relies on the sum of absolute difference (SAD), which minimises the effect of large outliers in a search area (Konolige 2010), in relation to the error in block comparison to find a minimum and suggest the most accurate match other comparisons are available such as sum of squared difference and normalised cross correlation which perform slightly differently particularly with block size is small (Ambrosch et al. 2007). In regions of optical uniformity or with areas of repeating patterns along epipolar lines it can be clearly assumed that errors in this matching function will yield a low-quality disparity map. It is therefore important that the environment is optically interesting as with most machine vision applications. Importantly the stereo matching variables are only applicable to one image set and as the target or environment change further tuning is required.

Depth calculation is a simple geometric triangulation after the creation of a disparity map that relies heavily on the object depth and the focal length and baseline of the camera setup. The detection of change of depth or depth resolution is given by the equation below for disparity calculated as the change in position of the subject between the left and right image(Δz), depth(z), the distance between the cameras (b), the focal length f and the change in distance from the stereo setup(Δd).

$$\Delta z = \frac{z^2}{f \cdot b} \Delta d$$

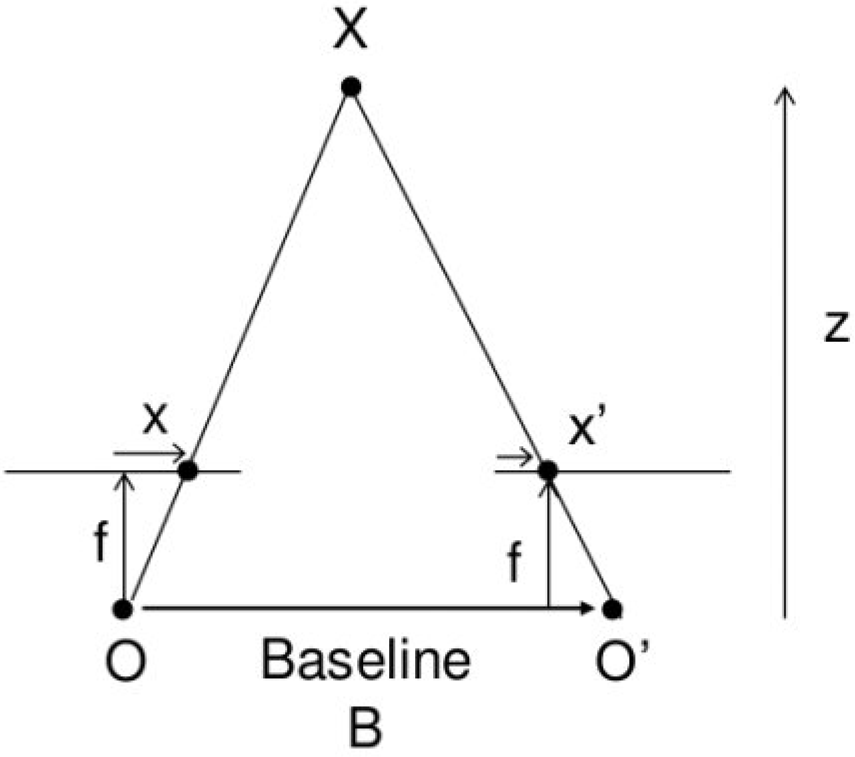


Figure 6 - Depth calculation from stereo disparity (OpenCV 2022)

In order to maximise the baseline within reason or the s500 platform so the stereo camera system would not impair the movement or risk damage to the camera or UAS. High resolution cameras to increase the ability for differentiation and disparity calculation. The Arducam Stereo Camera Hat outputs a composite image from the two cameras at a resolution 4056x3040 pixel image containing the left and right image 2028 x 3040 image per side. The general approach is listed in stages below:

- Take a stream of images and use the read() operation to select the image
- Use the calibrated corrections to remap and undistort
- Make it greyscale to be used for stereoBM
- Use StereoBM because it’s quick and decently accurate also stereoSGM which is more accurate and only a little bit more expensive as an option to prepare the disparity map
- Use the focal length, and baseline to calculate the depth.
- Combine the Optical flow depth assessment
- Dense optical flow on a ROI around the commanded delivery point

Modify the approach to elongate the translation that is perpendicular to the approach vector. This should increase the accuracy of data from depth from optical flow. A fixed height approach should satisfy this requirement.

Blend using EKF, this should ensure that there is some credulity to the data.

If the distance measured is less than the GPS distance to the point by the minimum variation calculated by the optics then the decision to alter the delivery method is made.

The continually calculated release point is calculated as a function of the vertical and horizontal speeds, the altitude and the payload variables returning the distance from the target required for release. During a cruise release the GPS position will trigger a delivery rather than the stereo system.

When modified to a hover delivery when the UAS arrives at the GPS position of the target the payload is released then the mission reverts to the cruise delivery for the next point.

Regulatory restrictions of UAS operation detailed by CASA are detailed below and will inform the deployment and testing phases of this implementation. The controlled environment of the testing areas will mitigate the risks as defined in Appendix 15.

Information regarding the operation and registration of UAS is defined through the CASA regulations (CASA 2021). A number of points are to be observed for UAS flights;

- UAS must be registered for research and development purposes.
- Must use a remote pilot licence or an RPA operator accreditation. While this design doesn't consider night flying other registration and requirements are in place in accordance with the CASA beyond visual line of sight (BVLOS) Limits as to
- Flying within 30m of people (not closer than 15m) must have a remote piloted operator's certificate (ReOC) and a remote Pilot Licence (RePL).
- To operate within 15m weight under 150kg, consent from personnel, minimise safety risks, document practices (RPAS operations manual), no closer than 15m do not fly above any person.
- All personnel with 30m must give consent.
- Dual redundant battery systems with mountings
- Proven ability to fly safely with 1 motor inoperative with maximum take-off weight
- GPS hold and return home function with 7 GNSS satellites
- Not within 5.5km (3nm) of controlled aerodrome
- BVLOS operation requires RPA flight auth, include risk assessments, complete a flight assessment

3.3 Design of test

The following breaks down the expected iterative testing process to prove and assess each stage of the implementation.

Test 1:

Assess the stereo algorithms in the creation of the disparity maps. Use variable Middlebury Dataset images; artroom, ladder and pendulum.

The disparity map will be assessed qualitatively to assess the clarity and discrimination of the subject and how it compares to the truth table provided with the image pairs.

- Density and discrimination
- Speed of changes made through tuning indicating calculation of a single frame.

    - The values of the variables to be used for similar scenes during UAS testing.

- These are rectified HD (1920x1080) resolution images that output a robust disparity map using the OpenCV stereo algorithms when they are optimally tuned.
- The subjects affect the variables that are used. Close detail as in artroom displays a poor disparirty map when attempting to observe the close detail this is acceptable in that the targets are at a range meaning that large disparities, which indicate a close subject, is not what this system would use. To maximise the viewing window using a small, 3 by default, minDisparity which sets up the comparison pixel window and sacrifices close detail detection. The noise created by this sacrificed can be offset by increasing the uniquenssRatio.
- Optimal variables for depth resolution as seen in Ladder and Pendulum indicate that a lower minDisparity, a middling numDisparity, a small block size, a high uniqueness ratio and a very high P1, P2 to post process the disparity map.
- This ensures that the results reject noise and capture as much
- (genuine?trustable?dependable?) results while also rejecting noise.
- Transport these variables to a camera setup and test again.

Test 2:

Assess the cameras setup using the same variables that we identified in test 1.

Using the test rig on a marked and measured area looking at a chessboard calibration image on a box at ranges out to the maximum disparity detection and assessing the accuracy of the distance calculated.

- Density
- Speed (USB at varying resolutions)
- Distance accuracy
- I increased the GPU memory from standard 128 to 256MB that worked to capture full resolution could this be managed between disparity and image capture?

Test 3:

Outdoor setup and control of UAS decent. Using the stereo camera setup as an altimeter the UAS will hold a GPS position and descend through an arbitrary decision height. 30m descend through 20m to 10m. On the 20m detection the UAS will indicate that it has detected the distance (spin? RTL - success or fail landing pad)

- Observe success
- Assess accuracy via tlogs

Test 4:

Fly at an interlocking plastic brick wall. Designate a delivery point behind the wall to trigger a decision to hover or cruise delivery drop.

Fly straight past with no detection or hover with detection

- Assess success
- vary speed/height until failure

1. Utilising a closed testing area with enough space to safely operate the UAS and setup the testing required.
2. Use interlocking bricks approximately 30 x 10cm, a “Lego” style block to build a wall to act as the obstruction built to a height of approximately 2m at an initial distance from 3m from the delivery point.

3. GPS RTK to layout the positions for release, positioning of wall and the designated delivery position.
4. Setup a mission via ArduPilot to match the positions including secondary and tertiary delivery point closer to the obstruction to test the resolution of the depth measurement.
5. Setup video recording devices for the drop zone, wall and the UAS. Tripods to enable sufficient field of view for review.
6. Run the mission and assess the position of the dropped object and the change in delivery method.
7. Move the delivery position and remap the GPS position and change approach speeds 5, 10, 15 m/s
8. Review footage and assess the outcomes.

Chapter 4 - Implementation Discussion and Results

4.1 UAS frame

The frame has a payload carry weight of 4kg and a typical endurance 60 minutes and speed of 15m/s these values are suitable for a test bed and to carry and deploy the test payloads (tennis balls) for flight testing. The camera gimbal does not exhibit a large increase in weight given its manufacture from 30% infill PLA 3D print and the negligible weights of the hollow carbon fibre gimbal rod, camera mounts and gimbal servo. The cameras do present a slight increase to the weight and their eccentric loading on the gimbal rod has been managed through the design of the camera mounts to provide a slight downwards attitude to minimise the load on the servo when typically, over larger transit distances the servos will be horizontal as a function of the height and the ground distance to the target.

Given the performance of this frame is largely restricted because of the stereo processing speed the speeds and endurance of this frame are not directly assessed as a measurable indicator of this design. The fixture of the cameras on the gimbal rod are likely to exhibit modal vibration which may affect the position and calibration of the stereo setup.

After flight testing has occurred a reassessment of the calibration is to be carried out, further securing the cameras would reduce the requirement to recalibrate should the individual cameras not move relative to each other although the vibration may reduce the efficacy of the calibration and therefore the disparity during specific flight envelopes. Review of the flight profile versus the disparity calculation may elucidate the effects of the gimbal's modal vibration.

4.2 Stereo Vision Implementation

The stereo distance measurement order of operations is detailed below the detailed programs written to carry out this process are detailed in Appendices I – R.

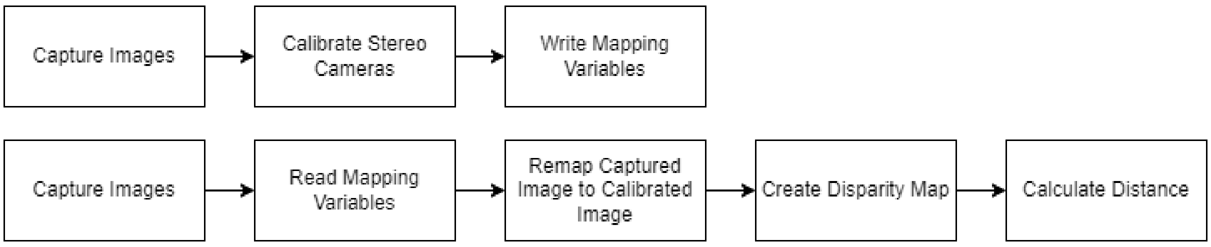


Figure 7 - Stereo Calibration (top) and Operating Operations (bottom)

4.2.1 Image Capture

The image capture occurs continuously throughout the algorithm using a video capture mode which enables the sensor to stay active in between shots to enable the highest frame rate over still capture. The camera synchronicity was tested by capturing images of a running stopwatch. On review of the captured image pairs, it was evident that the image pairs are not captured synchronously with the USB OpenCV camera read function yielding a 0.08s delay between cameras. After implementing the grab commands for both cameras and then utilizing the retrieve commands on the raw camera data, both of which are called during a single read command, the delay between image pairs was reduced to 0.03s between pairs. The image capture in the UAS implementation through the Arducam Stereo Cam Hat was tested in the same way and proved to be a synchronous capture of image pairs suitable use without correction.

The delay in captured image pairs when coupled with a translating camera system would have a large effect on the disparity map captured during this movement. The simple epipolar search of the StereoBM function, even after the calibration and rectification has taken place, would be unlikely to account for the y-axis misalignment of the image pairs and therefore would be unlikely to produce high quality disparity maps unless the block size was increased in order to account for this misalignment. The testing of the corrections required were outside of the scope of this implementation but research (Ambrosch et al. 2007)(Konolige 2010)(OpenCV 2022) has shown that an increased block size will result in a more computationally taxing matching algorithm if it would work at all.

Image capture resolution is a simple designation in the capture program but must be consistent between the calibration and stereo matching programs as discussed below.

The USB cameras used in the initial test build had auto focus enabled as a default setting which constantly readjusts the focal length to ensure the highest quality recording of images with varying subject depths, but the adjustment of focal length alters the intrinsic variables of the camera system therefore rendering the corrections and rectifications incorrect. OpenCV has functions to control a number of settings of the camera system, and while a number were utilized, attempts to disable the auto focus were unsuccessful via code. Open Broadcaster Software is an open-source video and streaming program which allows extensive customization of the streamed images. With this application the focal length could be fixed thus preserving the calibration data captured.

4.2.2 Calibration and Rectification Assessment

Calibration and rectification are the most important steps when attempting to employ stereo machine vision to interact with the outside world. The process was broken down into creation of an appropriate calibration target, capturing calibration images, calculating calibration variables. The rectification process remaps the images onto a common plane using the camera intrinsic and extrinsic variables. The processes followed here are heavily influenced by the tutorials published by OpenCV (2022) regarding this process, I have also taken influence and combined implementations from Nicolai Nielsen(2022) and LearnOpenCV (Sadekar 2021). The processes were incrementally deployed using the USB camera test bed this required variations to the approach and developed limitations discussed below.

OpenCV employs a chessboard, among other calibration targets, that can be customized prior to use. A 9 x 6 interior corner chessboard calibration pattern was used printed on an A4 sheet as shown in Appendix 1. This image must be displayed to the camera system on a single plane and it was taped to a cardboard sheet prior to any attempts to capture the images.

The literature review Hirschmuller (2007) cited a chessboard image held in landscape orientation covering 50% of the image and capturing at least 2 images provided the best calculation of intrinsic and extrinsic parameters and to ensure that all areas of the image area are captured with the chessboard calibration target in both camera images. The capture and calibration method used in this way adds image points to all areas of the camera field of view which results in a larger useable corrected image area.

This led to complexity when trying to balance the capture of the chessboard target in all corners of both images. If the size of the image is not large enough the calibrator program may not detect or worse incorrectly append the chessboard corners thereby returning erroneous calibration data resulting in poor rectification refer Appendix 3

In practice the original capture of the calibration target at a distance of approximately 2m resulted in no chessboard corners found from a 640x480 image from approximately 20 – 40 captured image pairs. After extensive testing varying the light, distance and captured resolution, up to 720p, no successful calibratable images were found. Adjusting the approach to calibration I utilized MATLAB’s Image Processing and Computer Vision module and the Stereo Calibrator App. Using this toolbox, I was able to consistently capture calibration image pairs and by designating chessboard corners calibration was successful for over 95% of captured images, refer Appendix 5

The reprojection error, or the accuracy of the corrections applied as a result of the calibration variables, resulted in approximately 5 pixels which is a poor result. The application of these corrections in the undistortion and rectification of test image stills to create a disparity map created exceedingly poor disparity maps and extensive tuning of algorithm variables could not return a useable disparity map.

Further testing revealed that one of the cameras had a significant distortion area in the center of the captured image which was a considerable departure from the expected image quality especially when compared to the matched camera pair.

After a new USB camera pair was purchased with a higher maximum resolution, 1080p, chessboard calibration images were captured again using the OpenCV methods and were successful. The calibration images were captured initially with a 0.08m baseline at a 640x480 pixel resolution and the find chessboard corners method captured successful and accurate corners for >95% of captured images. 40 images were captured and reprojection error was calculated at 0.02 pixels which is exceedingly accurate, but likely inaccurate. The results of the calibration and rectification can be assessed by reviewing the reprojection error of the calibrated images. The reprojection error when using 640x480 images is calculated at 0.04 pixels which is an excellent result indicating that the calculated intrinsic and extrinsic variables have been accurately to correct the calibration images. This in turn allows for accurate rectification resulting in captured images that represent the outside world. The result of this reprojection error was calculated following the tutorials from OpenCV (2022) but given the excellent result doubts were introduced as to the certainty of this result. Regardless the calibration, undisortion and rectification of the image pairs yielded good quality rectified images which were qualitatively compared using stereo anaglyphs prior to calculation of the disparity map.

For subsequent tests that extended the baseline to ~0.30m introduce further complications when balancing the detectable distance of the A4 calibration target, the distance from the camera system and the requirement to capture the calibration target in both image pairs in all areas of both images. Initially the increase in distance utilizing the low resolution caused failures in chessboard captures and incorrectly add image points through confusion of chessboard corners, refer Appendix 3.

The was as a function of the distance from the camera system and the captured image resolution so to balance the computational cost the calibration target was enlarged, initially to an A3 size for smaller baselines and eventually with the final implementation and the distance required given the baseline and the narrow field of view of the Raspberry Pi HQ camera lenses four A3 sheets were used to capture the calibration target at distances of approximately 5m as required. The result of the narrow field of view lead to a difficulty to capture all areas of the original camera image which reduced the calculated image points through the calibration process and therefore a reduced usable area available after undistortion and rectification.

The process of calibration and rectification as it is the most crucial stage of the depth calculation procedure also exposes the algorithm to the largest variance. Poor quality calibration images and poor technique in capture add cumulative errors since the assumption is that all captured images represent a true and accurate representation from which to calculate the extrinsic and intrinsic parameters.

Ground level testing demonstrated retention of the camera calibration variables without the need to recalibrate. The likelihood that recalibration is required after flight is high given the large amount of vibration likely to affect the gimbal assembly. Further, any changes in the camera system baseline, focal adjustment and image resolution all require a repetition of the entire calibration process prior to use.

4.2.3 Disparity Map

In initial testing attempts when reacting a disparity map in MATLAB following the calibration and rectification methods applied by the Stereo Camera Calibrator App. Utilising the captured calibration images as the reference images with which to create the disparity map and the rectification is applied to this image pair. A stereo anaglyph is produced to ensure that the distortion and rectification have been applied correctly before these rectified pairs are passed to the disparitySGM algorithm with some default parameters for disparity range and uniqueness factor. A qualitative assessment of the initial disparity maps even when varying the variables were very poor. Speckling was severe with little to no continuity along any detail from the reference images, Appendix 5.

Since calibration was difficult and returned poor disparity maps, attempts were made to create disparity map without calibration utilising object recognition through Oriented FAST and rotated BRIEF (ORB) matching through MATLAB's detectORBFeatures function as detailed by Rublee et al (2012). This is a simple concept where ORB features are matched between the left and right stereo image pairs and an estimated fundamental matrix is calculated allowing for a rudimentary rectification by aligning the matched SIFT features on the same epipolar lines. A disparity map can then be created after these points are aligned in the best estimate as per the disparitySGM algorithm. Figure 6 shows the result of the ORBmatching uncalibrated rectification disparity map result and represents a good approximation of disparity, even without tuning, because of the ORB feature rectification and therefore does represent a possible avenue for future work.

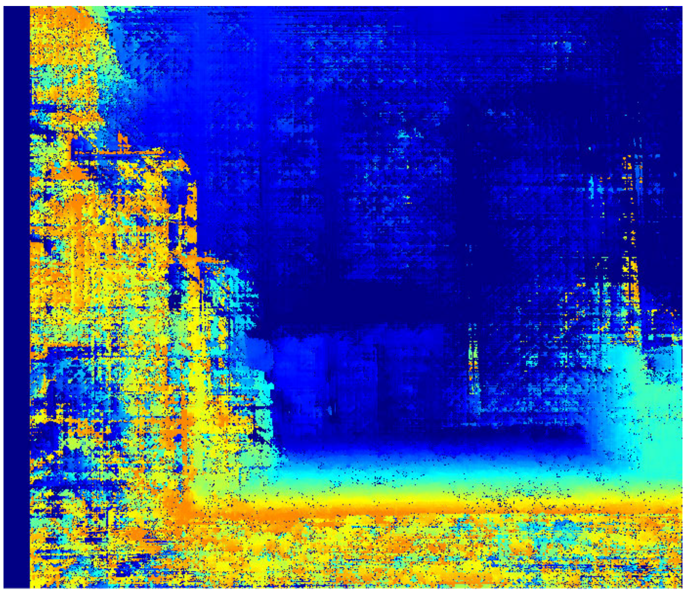


Figure 8 - Disparity Map After ORB Feature Rectification

After the new camera system was obtained OpenCV was used to produce disparity maps using the StereoBM and the StereoSGBM functions. Given the success of the calibration and rectification stages through OpenCV's implementation the disparity maps could be created and assessed. To being the implementation the StereoSGBM function was used, given its extended search patterns, in order to maximise the chance of a high-quality disparity map being produced if calibration and rectification had a higher reprojection error. Initially, default variables were used as a proof of concept and the disparity map was qualitatively assessed, the results were poor. To make the tuning of these disparity map variables more efficient and to gain some indication of the speed of processing a GUI was created (Sadekar, 2021). This GUI allowed for adjustment of the variables to tune the disparity map for qualitative assessment through a continuous loop and vastly increased to speed of the tuning process while also making clearer how each variable affected the map's quality. It was through this process that good quality maps were created and the key variables for varying image compositions were identified.

The variables that can tune the stereo matching and disparity map creation are detailed below. The alteration of these variables makes dramatic differences to the clarity and continuity of the disparity map, and the effective range of detection.

StereoBM variables

Block Size – How much of the image is attempted to be match with each pass

Disp12maxdiff – Defines the maximum difference in the comparison between matches going from right to left and from left to right.

Mindisparity – An initial offset from which to start the block match. This will enhance the disparity detected from subjects in the foreground but will often overlook element in the background where disparity is small.

Numdisparities – how far along the search area, usually the epipolar line, to translate the window in search of a match

Prefiltercap, Prefiltersize – Pre-processing variables that enhance the texture in a region and normalise the brightness of the image prior to block matching which increases the accuracy of the algorithm.

speckleRange, speckleWindowSize – Speckles are produced at the edges of subjects in the image or where the algorithm incorrectly attributes a match. These post-processing variables remove the speckle by ensuring that it is below a certain size or how distinct it is from its neighbours.

TextureThreshold – The minimum allowable texture from any region of the captured image. The area is rejected if under this threshold.

Uniquenessratio – A weighted comparison between the next best minimum match in the search algorithm in terms of percentage.

StereoSGBM variables

The variables for the semi global algorithm are the similar to the block matching model with the addition of below:

P1 – post processing penalty variable controlling immediately neighbouring pixels

P2 – post processing penalty variable controlling broader neighbouring pixels. P2 is greater than P1.

4.2.4 Middlebury Dataset Assessment

The Middlebury dataset as detailed in the literature represents a common set of images with which to assess the stereo algorithms accuracy in creating a disparity map. The results using the StereoBM algorithm using the rectified images supplied in the dataset yield a dense and accurate disparity map as seen in Figure 6. The tuning required between datasets is different given the change in the requirements of the range of the targets in the reference image. The StereoSGBM algorithm does provide a slightly denser disparity map but does not appreciable increase the quality of the result. This is likely because of the calibration and importantly rectification of the images that provides a greater chance of finding a match along epipolar lines. This result while proving the efficacy of both stereo disparity algorithms but also reiterates the importance of calibration and rectification stages of the routine.

The results from the Middlebury datasets in tuning to produce the densest disparity map with the clearest discrimination demonstrates the differences in variables required to capture disparity

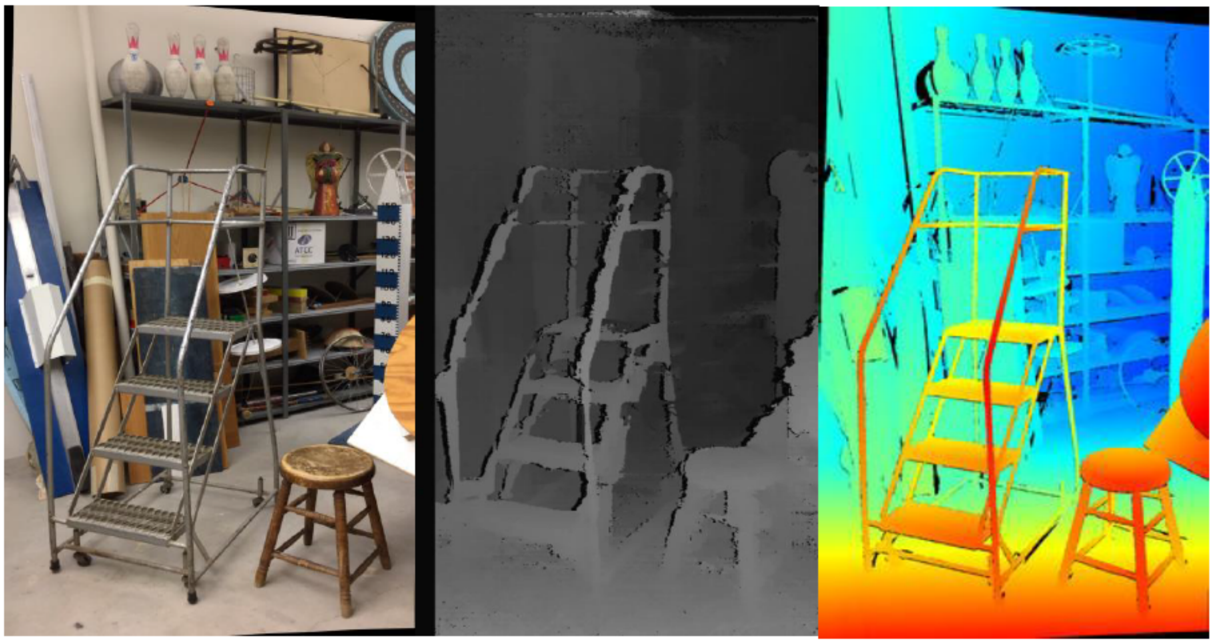
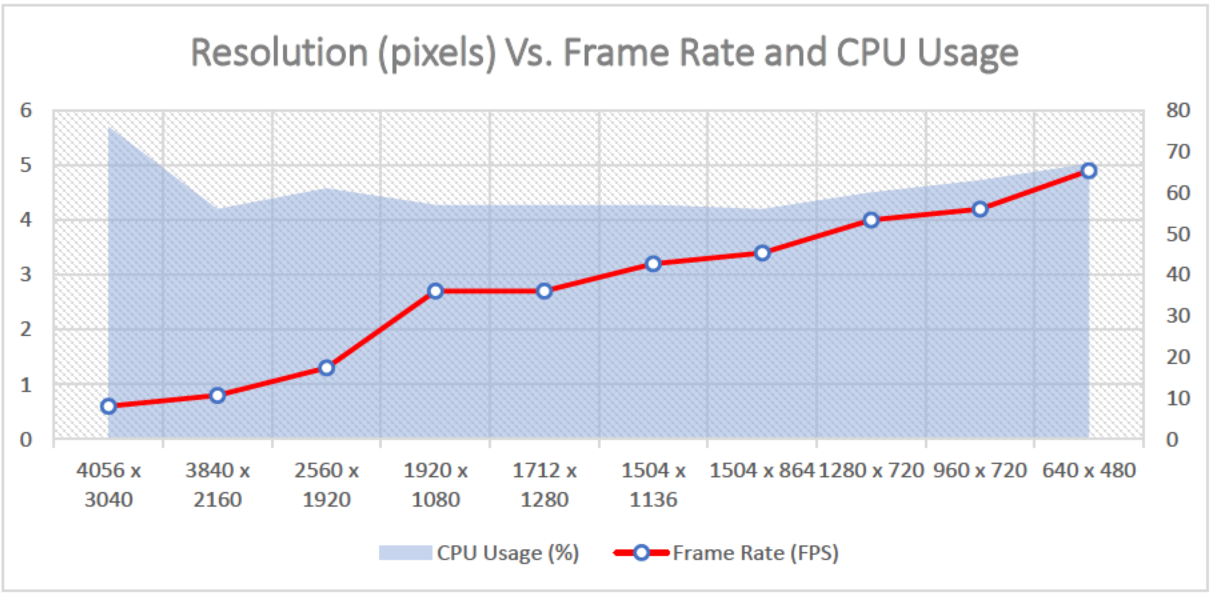


Figure 9 - Middlebury Dataset Reference, stereoBM Disparity Map and Truthtable

4.2.5 Stereo Algorithm

The stereoBM algorithm is computationally less expensive than the stereoSGBM given the difference in the search function to match the blocks. The stereoBM algorithm operates at approximately 1/5th the FPS and this can likely be attributed to the additional search functions of the SGBM search methods. Although the results are less dense and particularly in the real-world tests do not manage to find matches in areas of lesser visual complexity.

Table 3 - Resolution of captured image and the effect of frames per second and CPU usage



The resolution is a large factor to frame rates to capture the minimum resolution the system must capture an image of twice the width to then split into the image pairs. This means that rather than a single camera capturing a smaller image the system synchronously captures a larger image so even though the resulting image pairs are of a lower resolution, the captured images still tax the companion computer as if it was a larger resolution image. This is a trade-off between synchronicity and computational cost. While the result of asynchronous image capture hasn't been reviewed in this implementation. Maximum resolution available for still images is 4056 x 3040 pixel with a standard resolution retaining the same aspect ratio of 1.33 or 1.77.

The systems performance indicates that the CPU usage is not directly tied to the FPS, while there is a trip point where increasing the captured resolution does begin to affect the CPU usage the baseline of the program seems to be 61% CPU usage with the stereo algorithm only. Increased memory allocated to the GPU does allow the system to capture higher resolution images but there is a clear reduction in the frame rates which is directly correlated to the resolution. The largest resolution with the smallest CPU usage and the highest frame rate occurs at the 1504 x 1136 image resolution and will be adopted for this test. Unfortunately, this does indicate that the Raspberry Pi HQ cameras are not the optimum hardware for this application as the computational cost and frame rate does not offer a larger resolution of captured image as was hypothesized, especially not enough for UAS applications.

Thermal effects of the computation push the temperature to 75°C in a temperature-controlled office at 20°C. Raspberry Pi operates at a maximum temperature of 85°C and begins to throttle performance at 80°C. It is likely that in operation in an outdoor environment the likelihood of an increased temperature and thermal throttling further decreasing the companion computers operation.

Image pyramids to assess the differences in the stereo disparity created. On the ladder dataset by pyramiding down reduces the resolution of the image, given the pre-rectified nature of the supplied image pairs there is no requirement to recalculate any extrinsic of intrinsic variables to apply the stereo disparity calculations. The results demonstrate a smooth but diminishingly detailed map when compared to the original 1080x1920 pixel image.

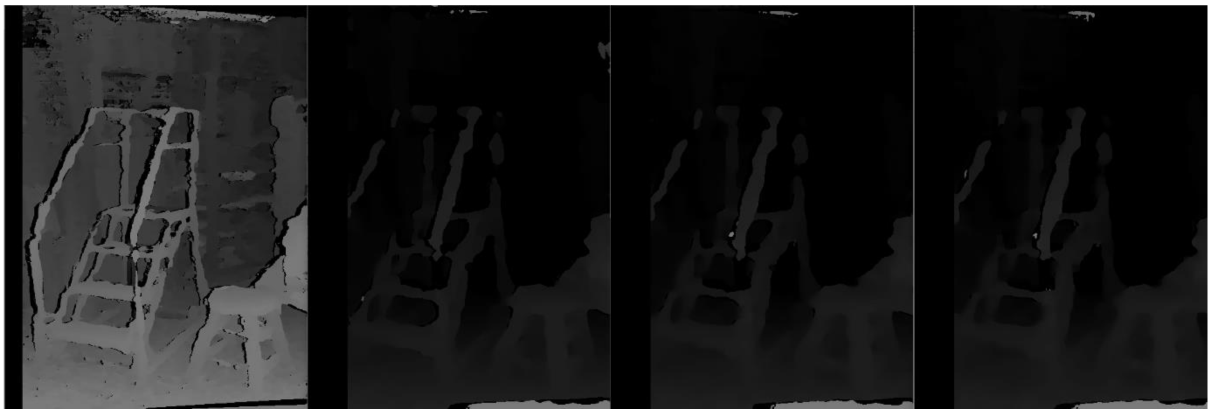


Figure 10 - Image Pyramid Disparity Assessment. Full Resolution, Half Resolution, Quarter Resolution, Eighth Resolution from Left to Right

The results of these tests demonstrate the increase in disparity as a function of resolution which was the minor hypothesis of this research. This evidence supports the hypothesis and reinforces the decision to attempt to implement high resolution stereo disparity as an effective approach. It is clear from the research and the implementation thus far that the increase in resolution should be optimised against the run time of the stereo disparity loop.

4.2.6 Depth Measurement and Accuracy

In the second stage of testing where the calibrated stereo setup is used to measure distance in a real-world environment the results proved an acceptable accuracy for detected disparity. The measurement in an interior environment with fluorescent lighting given a 640x480 pixel captured image with a 0.08m baseline and a 56.6° horizontal field of view. The target was moved throughout the testing intervals as described in the table below.

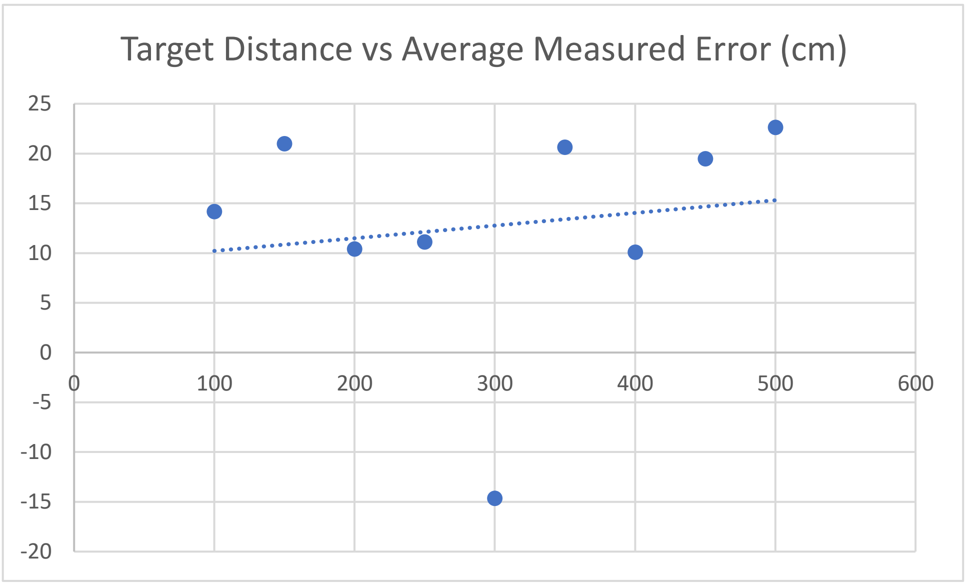


Table 4 - Distance Measurement Results. 0.8m baseline 5m target 1m intervals

target distance (cm)	measurement 1	2	3	4	avg measure	avg error
100	86.4	84.4	85.3	87.2	85.825	14.175
150	130.6	128.9	127.3	129.3	129.025	20.975
200	193.2	187.2	189.6	188.4	189.6	10.4
250	243.8	238.7	235.5	237.6	238.9	11.1
300	307.4	317	310.5	323.7	314.65	-14.65
350	340.2	338.1	315.4	323.8	329.375	20.625
400	389.5	391.2	388.3	390.7	389.925	10.075
450	423.7	432.7	430.1	435.6	430.525	19.475
500	486.9	457.6	488.2	476.8	477.375	22.625

Figure 11 - Initial StereoSGBM Distance Measurement 0.08m Baseline 640x480 Image

The results indicate a consistent accuracy when disparity is successfully mapped to a tolerance far smaller than is required by the delivery method. The variance in the disparity can measure a change in distance of 20mm. The accuracy of the results does indicate a consistent under reading of approximately 12.8mm that is consistent across the detection range. The result of this error could be explained through the assumptions of the geometry and the accuracy of the output as a function of the measured baseline, horizontal field of view and distance to the target. This does indicate that the calibration stage is vitally important to the measurement of distance as well as the pure collection of data.

This interim test had limitations calculating disparity and identifying the target, the calibration chessboard, at distances greater than 5m with the baseline of 0.08m. While the test provided a proof of concept the limited distance is assumed to be as a function of baseline which further tests prove.

4.2.7 Stereo Outdoor

The outdoor stereo implementation when using the stereoSGBM algorithm showed a sparse disparity map with a particularly strong disparity indication along the centre of the image. The increase in baseline from initial testing at 0.08m to 0.26m provided a sizeable increase in the disparity map detection range with positive detections out to approximately 40m. The algorithm performs well in identifying obvious and optically interesting targets and shows strong disparity indications and discrimination for these objects. This result indicates that the likelihood of capturing mid distance (10-40m) obstructions is high and when the disparity is found that distance measurement can be inferred but what not pursued in this test.

The frame rates and calculations occur at a speed that clearly track obstructions particularly objects with large optical differences between surrounding objects and without complex repeating patterns as with typical machine vision applications.

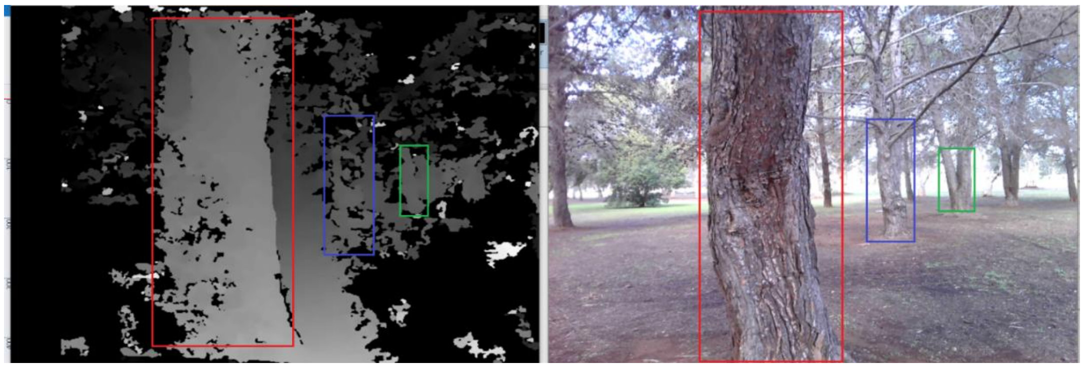


Figure 9 - StereoSGBM 0.26m Baseline 640x480 Disparity and Differentiation

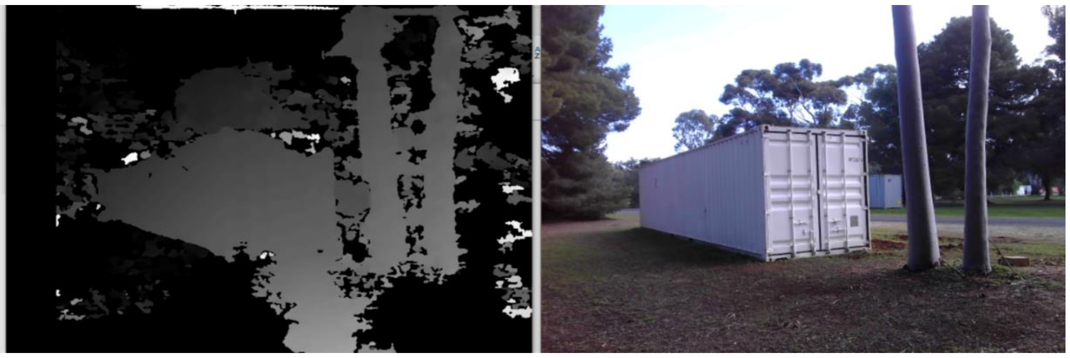


Figure 10 - Outdoor stereoSGBM Algorithm Results 640 x 480 Resolution 0.26m Baseline

4.3 UAS Based Stereo

4.3.1 StereoSGBM Algorithm

After initial success with the disparity maps calculated with the USB camera test setup OpenCV's semi global block matching algorithm was implemented using the UAS companion computer and the Raspberry Pi HQ cameras capturing images through the Arducam Stereo Hat.

This implementation demonstrated far slower frame rates with exceedingly high CPU usage, measured with the "htop" function. This function returned an average CPU utilisation of 85% on average with considerably higher operating temperatures 75°C in a climate-controlled workshop space at 18°C. The Raspberry Pi system has a two-stage thermal protection function where soft throttling occurs at 80°C and significant throttling of the CPU occurs at 85°C. The likelihood that this temperature would be reached is unknown given the added complexity of the heat sink and their effect of the thermal protection while in flight. The CPU usage however while not running the autopilot program proves that this algorithm is far too computationally expensive and could not be appropriately balanced or tuned for operation.

The StereoSGBM method, while it produced far better disparity maps at distances closer to the operating requirements in the original testing setup with a laptop and USB cameras, fails to run at the required FPS only achieving 1FPS average. This FPS is too low to be used with a UAS especially when decision making distances are critical and the underlying aim of improving efficiency by maximising operating speed.

4.3.2 StereoBM Algorithm

The implementation of the UAS stereo system using the StereoBM method. The results vary and the tuning to focus the horopter to maximise the detected distance proves good detection but reduced fidelity i.e., the disparity can no longer differentiate approximately 0.20m at 15m but does produce strong indications that can measure distance accurately 10%. This error seems constant through the detected ranges after tuning therefore it is likely that the errors are cumulative considering the cameras stated focal length, FOV_H and the specific variables tuned which can significantly affect the disparity detected.

In field trials the horopter using the stereoBM algorithm is far more restricted than the stereoSGBM algorithm. This means that disparity is not detected when objects are too close or too far away from the stereo camera setup. This viewing band close the horopter is 7m deep and calculates clear gradients in the disparity which can measure distance. This occurs as a function of the stereo parameters primarily the numdisparities and the mindisparities variables. These variables can synthesise a baseline using the mindisparities variable to setup the minimum search distance in the image plane for matching.

The distance measurement is very accurate when taken along the horopter results are within 0.50m at 15m 3.3% error but taking measurements closer or further from this distance e.g., along a plane that is oblique to the image plane provides a rapidly changing disparity and therefore increasing error when observing non-crossing and crossing disparities. It is unclear from the qualitative assessment whether this disparity changes exponentially.

By designating a region of interest close to the centre of the image plane the errors relating to this changing disparity could be minimised although through reducing the effective area reduces the ability to scope every obstruction.

4.4 Optical Flow Results

The results were poor using dense optical flow and OpenCV the frame rates using a 640x480 image were approximately 5FPS, but the effective distance and disparity was poor see Appendix E. The effective range tested demonstrated discernible differences in flow over approximately 1m from the camera system. Given the deviation from the implementation scope as a result of the pursuit of stereo vision implementation as a priority further development of the optical flow algorithm were not pursued as a viable contributor to the final distance calculation. Sparse flow could be used but does raise the risk of missing transitory obstacles or obstacles that are not sufficiently optical interesting or dissimilar to their environment.

4.4.1 Data Blending

Data blending was not attempted during this implementation given the unusable results from the initial optical flow tests and the limitations of scope when attempting to prioritise stereo vision.

Chapter 5 - Future Work

This research and particularly implementation has been illuminating in separating the theory and application of machine vision deployment. The resolution of the captured images was the limiting factor in terms of frame rate and accuracy and further work would be appropriate to refine this implementation. The collection of data in the function of high-resolution image represents an inefficient allocation of resources and given the limitations of the companion computers where a more intelligent use of resources and data is applied. While the combination of data could be a viable option whether that is optical flow or depth cues.

Marker emplacement or object detection would reduce the requirement for dense stereo and dramatically reduces the computational cost after that marker has been identified. This approach would reduce the computational overhead and increase the efficacy of CNN and sparse optical flow implementations.

Similarly, the ability to train a complex, complete model offboard from the UAS system and implement it with variable “light weight” machine learning applications e.g. TFMic. The implementation of complete machine learning models is very efficient and may be a valid approach although the training element would be large given the various environments it would be applied in.

Another avenue could be to utilise higher powered GPU or FGPA driven hardware as a focus could be a solution to the limited companion computer hardware. Since the simplistic matrix operations representing block matching and disparity calculation a GPU would be more suited to stereo applications, this is supported in the literature. Similarly, offboard computation, mobile edge computing and threading for image capture and assessment could improve the frequency of the algorithm but the approach to increase the computational power of the system and not approach the data from a more refined position would be inefficient when compared to the other approaches mentioned here and in the literature.

Chapter 6 - Conclusions

The research has demonstrated the basics of stereo vision applications and highlights the variables critical to the complex computation of depth from a stereo machine vision arrangement.

The hardware selected to accurately represent common companion computers proved to be underpowered for the application and did not produce a suitable disparity map to be used with a UAS, particularly up to the speeds and over distances required for operation. The stereo setup could be utilised in an indoor rover where speed and ranges would both be reduced as well as the ability for higher computational power as it relates to weight.

The stereo algorithm has been demonstrated to be effective when the algorithms are executed on a typical laptop computer on pre-rectified image pairs from the Middlebury Stereo data sets and in ground-based trials. This method demonstrated detailed and accurate disparity maps from which depth could be accurately inferred. The initial field test application utilising USB cameras and the laptop demonstrated detailed disparity maps out to a range of approximately 40m at a frame rate of approximately 10 FPS with an accuracy of approximately 5m. This build demonstrated the proof of concept to a point where implementation on a UAS was pursued.

The second iteration of the camera system, while having the ability to capture images of high resolution, failed to capture these resolutions at a frame rate required for UAS operation when considering the computational cost of disparity mapping with the semi global search algorithm. The additional computational weight of the stereo rectification and disparity calculation and also the requirement to interact with the UAS autopilot meant that the system was not produce the frame rate required and the algorithm was changed to a less intense block matching algorithm.

This iteration after tuning provided sparse but accurate disparity and distance measurements at a range of 20m with a steady offset error of approximately 0.3m likely because of the inconsistencies with the manufacturer's specifications regarding focal length, horizontal field of view and the accuracy of the measured baseline. The algorithm produced an approximate 5 FPS capturing 640x480 pixel images. CPU usage was measured at an average of 60% and kept system temperature below the threshold of thermal throttling at an approximate 60°C at 20°C ambient temperature, this operating temperature would likely be reduced once airflow is available for cooling.

Further, the implementation of the optical flow element was unsatisfactory with poor accuracy and a minimal effective range of 0.5m.

While the hypothesis was supported through the image pyramid testing in that a greater disparity is detected with a larger resolution. The computational cost when capturing and assessing larger resolution images is prohibitive especially when using a low power companion computer in the Raspberry Pi 4 (4GB). It is clear that the implementation is not appropriate for UAS deployment given the limitations of the hardware and the computational cost of the fairly generic dense stereo vision algorithm and more importantly the practical limitations of the physical implementation against the theoretical approach led to extensive reconsiderations of approach, replacement of components and severely limited the scope of this implementation. The future work suggested in this research demonstrates the myriad of other approaches focusing on a more efficient use of the data captured rather than an increase in the data itself or an increase in computational power. This is a key factor in applying complex machine vision applications to lighter computing tools and microcontrollers.

Reference list

Alam, I 2020, ‘The Advancements of Computer Vision: How Is It Used Today?’, *Techserious*, viewed 29 Sep 2022, <https://techserious.com/computer-vision/&imp=46959>

Ambrosch, K., Kubinger, W., Humenberger, M. and Steininger, A., 2007, June. Hardware implementation of an SAD based stereo vision algorithm. In *2007 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-6). IEEE.

Australian Institute of Health and Welfare 2022, ‘*Rural and remote health*’, viewed 20 Aug 2022, <https://www.aihw.gov.au/reports/rural-remote-australians/rural-and-remote-health#Profile%20of%20rural%20and%20remote%20Australians>

Baraldi, Patrizia & De Micheli, Enrico & Uras, Sergio, 1989, Motion and Depth from Optical Flow. 10.5244/C.3.35, viewed 03 Mar 22, available at https://www.researchgate.net/publication/239760421_Motion_and_Depth_from_Optical_Flow

Bell, R 2022, ‘Last Mile Delivery Explained: Trends, Challenges, Costs & More’, *Merchants Fleet*, viewed 20 Jul 2022, < https://www.merchantsfleet.com/industry-insights/what-is-last-mile-delivery/#:~:text=What%20Is%20the%20Last%20Mile%20Problem%3F%20The%20last,and%20customer%20expectations%2C%20just%20to%20name%20a%20few.>

Birchfield, S, Tomasi, C 1999, “Depth discontinuities by pixel-to-pixelstereo”, International Journal of Computer Vision, vol. 35, no. 3,pp. 269–293.

Bouzid, Y., Bestaoui, Y. and Siguerdidjane, H., 2017, September. Quadrotor-UAV optimal coverage path planning in cluttered environment with a limited onboard energy. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 979-984). IEEE.

Bowditch, G 2020, ‘A COVID-19 wake-up call: Get smarter about population density’, *Financial Review*, viewed 04 Jul 2022, < https://www.afr.com/companies/infrastructure/a-covid-19-wake-up-call-get-smarter-about-population-density-20200415-p54k09>

Brox, T, Bruhn, A, Papenberg, N, Weickert, J 2004, High Accuracy Optical Flow Estimation Based on a theory for Warping, Mathematical Image Analysis Group Faculty of Mathematics and Computer Science Saarland University, Building 27, 66041 Saarbr¨ucken, Germany, viewed 31Mar22, available at < https://link.springer.com/content/pdf/10.1007/978-3-540-24673-2_3.pdf>

Chang, J.R. and Chen, Y.S., 2018. Pyramid stereo matching network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5410-5418).

Cherry, K 2021, Monocular Cues for Depth Perception, Verywell Mind, Cognititve psychology, viewed 25 Mar 22, available at <https://www.verywellmind.com/what-are-monocular-cues-2795829>

Civil Aviation Safety Authority 2022, ‘Drone delivery services’, *Industry Initiatives*, viewed 10 Apr 2022, <https://www.casa.gov.au/drones/industry-initiatives/drone-delivery-services>

Civil Aviation Safety Authority 2022, Drones, *Commonwealth of Australia*, viewed 28 Feb 22, <https://www.casa.gov.au/drones>

Civil Aviation Safety Regulation 1998 (Cwlth)

Cooper, James, Mihailo Azhar, Trevor Gee, Wannes Van Der Mark, Patrice Delmas, and Georgy Gimelfarb. "A raspberry pi 2-based stereo camera depth meter." In 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), pp. 274-277. IEEE, 2017.

Debnath, S.K., Omar, R. and Latip, N.B.A., 2019. A review on energy efficient path planning algorithms for unmanned air vehicles. *Computational Science and Technology*, pp.523-532.

Department of Infrastructure, Transport, Regional Development and Communications 2022, Managing Drone Noise, viewed 14 Mar 22, available at <https://www.infrastructure.gov.au/infrastructure-transport-vehicles/aviation/emerging-aviation-technologies/managing-drone-noise>

Diaz, C 2022, '*Five ways to solve last mile problems in 2022*', Netlogistik, viewed 10 Jul 2022, < https://www.netlogistik.com/en/blog/5-ways-to-solve-last-mile-problems-in-2022>

Ergene, Y., 2016. Analysis of unmanned systems in military logistics. Naval Postgraduate School Naval Postgraduate School United States.

Eser, A, 2020, The Depth I: Stereo Calibration and Rectification, Python in Plain English, viewed 15 Mar 2021, available at: https://python.plainenglish.io/the-depth-i-stereo-calibration-and-rectification-24da7b0fb1e0#:~:text=Rectification%20is%20basically%20calibration%20between%20two%20cameras.%20If,camera%20and%20P2%20%28x2%2Cy%29%20for%20the%20second%20camera.

European Cockpit Association, 2020. Unmanned Aircraft Systems and the Concepts of Automation and Autonomy. ECA Briefing Paper, pp.1-7.

Galvez, R.L., Dadios, E.P. and Bandala, A.A., 2014, November. Path planning for quadrotor UAV using genetic algorithm. In *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)* (pp. 1-6). IEEE.

Gehrig, S, Eberli, F, Meyer, T 2009, October. A real-time low-power stereo vision engine using semi-global matching. In International Conference on Computer Vision Systems (pp. 134-143). Springer, Berlin, Heidelberg.

Hariyama, M., Yokoyama, N. and Kameyama, M., 2008. Design of a trinocular-stereo-vision VLSI processor based on optimal scheduling. IEICE transactions on electronics, 91(4), pp.479-486.

Hariyama, Masanori & Yokoyama, Naoto & Kameyama, Michitaka. (2008). Design of a Trinocular-Stereo-Vision VLSI Processor Based on Optimal Scheduling. IEICE Transactions on Electronics. 91-C. 479-486. 10.1093/ietele/e91-c.4.479.

Hamrouni, S., Louhichi, H., Aissia, H.B. and Elhajem, M., 2012. A new method for stereo-cameras self-calibration in Scheimpflug condition. In *15th international symposium on flow visualization* (pp. 1-10).

Heikkilä, J. and Silven, O., 1997: A four-step camera calibration procedure with implicit image correction. CVPR97

Hirschmüller, H 2005, 'Accurate and efficient stereo processing by semi-global matching and mutual information', IEEE Conference on Computer Vision and Pattern Recognition. pp. 807–814.

Hirschmuller, H 2007, 'Stereo processing by semiglobal matching and mutual information' *IEEE Transactions on pattern analysis and machine intelligence*, *30*(2), pp.328-341.

Intel Corporation 2022, 'Stereo Depth', *Intel Real Sense*, viewed 15 Mar 2022, <https://www.intelrealsense.com/stereo-depth/>

Javidnia, H. and Corcoran, P., 2017. Accurate depth map estimation from small motions. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 2453-2461).

Konolige, K., 1998. Small vision systems: Hardware and implementation. In Robotics research (pp. 203-212). Springer, London.

Kumar, K, Desai, U 1994. New algorithms for 3D surface description from binocular stereo using integration. Journal of the Franklin Institute, 331(5), pp.531-554.

Kumar, P 2022, ‘MIPI Cameras vs USB Cameras: a Detailed Comparison’, *Blog Posts, e-con Systems, Industrial Vision (Computer Vision), Sensors*, weblog post,24 Jan 2022, viewed 03 Mar 22, <https://www.edge-ai-vision.com/2022/01/mipi-cameras-vs-usb-cameras-a-detailed-comparison/>

Louhichi, H., Fournel, T., Lavest, J.M. and Aissia, H.B., 2007. Self-calibration of Scheimpflug cameras: an easy protocol. *Measurement Science and Technology*, *18*(8), p.2616.

Macrotrends 2022, ‘Australia Population Density 1950-2022’, *Macrotrends*, viewed 07 Jul 22, <https://www.macrotrends.net/countries/AUS/australia/population-density#:~:text=The%20current%20population%20density%20of,a%200.98%25%20increase%20from%202020.>

Mamdouh, T 2020, ‘Color spaces (RGB vs HSV) - Which one you should use?’, *HubPages*, viewed 13 Apr 2022, < https://discover.hubpages.com/technology/Color-spaces-RGB-vs-HSV-Which-one-to-use>

MathWorks 2022, *Stereo Search Function*, [Online]. Available at URL https://au.mathworks.com/help/visionhdl/ug/stereoscopic-disparity.html?searchHighlight=stereoSGBM&s_tid=srchtitle_stereoSGBM_1.

Miangoleh, S.M.H., Dille, S., Mai, L., Paris, S. and Aksoy, Y., 2021. Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9685-9694).

Nielsen, N 2022, ‘Computer Vision’, *Robotics - Computer Vision, Deep Learning and Artificial Intelligence*, viewed 30 Apr 2022, <https://github.com/niconielsen32>

Nvidia 2022, ‘Jetson TX2 NX Module’, *Nvidia Developer*, viewed 07 Jul 2022,<https://developer.nvidia.com/embedded/jetson-tx2-nx.

NYU, Szeliski, R, Lazebnik, S, Seitz, S, Efros, A, Liu, C, Durand, F 2012, ‘Lecture 13 Optical Flow’, viewed 22 Mar 22, <https://cs.nyu.edu/~fergus/teaching/vision_2012/13_opticalflow.pdf>

O’Donovan, P 2005, Optical Flow: Techniques and Applications, *The University of Saskatchewan*, viewed 17 Mar 22, <http://www.dgp.toronto.edu/~donovan/stabilization/opticalflow.pdf>

OpenCV 2022, ‘Camera Calibration and 3D Reconstruction’ , *OpenCV 4.6.0*, viewed 20 Mar 2022, <https://docs.opencv.org/4.6.0/d9/d0c/group__calib3d.html>

OpenCV 2022, Depth Map from Stereo Images, image, viewed at https://docs.opencv.org/4.x/dd/d53/tutorial_py_depthmap.html

Pani, A., Mishra, S., Golias, M. and Figliozzi, M., 2020. Evaluating public acceptance of autonomous delivery robots during COVID-19 pandemic. Transportation research part D: transport and environment, 89, p.102600.

Perez, T., Williams, B. and de Lamberterie, P., 2012. Evaluation of robust autonomy and implications on UAS certification and design. In Proceedings of the 28th Congress of the International Council of the Aeronautical Sciences (pp. 1-9). Optimage Ltd./International Council of the Aeronautical Sciences-ICAS.

Pettigrew S, Fritschi L, Norman R. 2018, The Potential Implications of Autonomous Vehicles in and around the Workplace. Int J Environ Res Public Health. 15(9):1876. Published 2018 Aug 30. doi:10.3390/ijerph15091876

PiAustralia 2022, Raspberry Pi Starter Kit, *Raspberry Pi Starter Kit*, viewed 03 Feb 2022,< https://raspberry.piaustralia.com.au/products/raspberry-pi-starter-kit#4gb>

Placek, M, 2022. What was the approximate weight of this particular purchase?, viewed 13 jun 22, available at: https://www.statista.com/statistics/974065/cross-border-delivery-package-weight-worldwide/#:~:text=This%20statistic%20shows%20the%20results%20of%20a%20global,between%200.2%20kg%20to%200.5%20kg.%20Read%20more

Pollefeys, M., Koch, R. and Van Gool, L., 1999, September. A simple and efficient rectification method for general motion. In *Proceedings of the Seventh IEEE International Conference on Computer Vision* (Vol. 1, pp. 496-501). IEEE.

Qi, W., Li, F. and Zhenzhong, L., 2010, May. Review on camera calibration. In *2010 Chinese Control and Decision Conference* (pp. 3354-3358). IEEE.

Quénot, G., Rambert, A., Lusseyran, F. and Gougat, P., 2001, September. Simple and accurate PIV camera calibration using a single target image and camera focal length. In *4th international symposium on of particle image velocimetry. Springer, Göttingen, Germany* (pp. 17-19).

Rana, K., Praharaj, S. and Nanda, T., 2016. Unmanned Aerial Vehicles (UAVs): An Emerging Technology for Logistics. International Journal of Business and Management Invention, 5(5), pp.86-92.

Ranieri, L., Digiesi, S., Silvestri, B. and Roccotelli, M., 2018. A review of last mile logistics innovations in an externalities cost reduction vision. *Sustainability*, *10*(3), p.782.

Remondino, F. and Fraser, C., 2006. Digital camera calibration methods: considerations and comparisons. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *36*(5), pp.266-272.

Rodrigues et al., 2022, Patterns 3, 100569, August 12, 2022 ª 2022 The Author(s), viewed 12 August 2022, <https://doi.org/10.1016/j.patter.2022.100569>

Rodrigues, T.A., Patrikar, J., Oliveira, N.L., Matthews, H.S., Scherer, S. and Samaras, C., 2021. Drone flight data reveal energy and greenhouse gas emissions savings for small package delivery. arXiv preprint arXiv:2111.11463.

Rodrigues, T.A., Patrikar, J., Oliveira, N.L., Matthews, H.S., Scherer, S. and Samaras, C., 2022. Drone flight data reveal energy and greenhouse gas emissions savings for very small package delivery. Patterns, 3(8), p.100569.

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G., 2011, November. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision* (pp. 2564-2571). Ieee.

Sadekar, K 2021, 'Depth Estimation using Stereo Camera and OpenCV (Python/C++)', *LearnOpenCV*, viewed 16 Feb 2022, <https://learnopencv.com/depth-perception-using-stereo-camera-python-c/>

Saxena, A., Schulte, J. & Ng, A., 2008. Depth Estimation Using Monocular and Stereo Cues, Stanford, CA, USA: Computer Sceince Department, Standford University.

Scharstein, D, Szeliski, R, Hirschmüller, H 2022, 'Middlebury, Stereo Vision Page',*vision.middlebury.edu*, viewed 29 Apr 2022, < https://vision.middlebury.edu/stereo/>

Scharstein, D, Hirschmüller, H, Kitajima, Y., Krathwohl, G, Nesic,N., Wang, X, and Westling,P, 2014. High-resolution stereo datasets with subpixel-accurate ground truth. In German Conference on Pattern Recognition (GCPR 2014), Münster, Germany

Shao, P.C., 2020. Risk assessment for UAS logistic delivery under UAS traffic management environment. Aerospace, 7(10), p.140.

Simpson, W., 1993. Optic Flow and Depth Perception, s.l.: Spat Vis. 1993;7(1):35-75. doi: 10.1163/156856893x00036. Erratum in: Spat Vis 1993;7(2):199. PMID: 8494808..

Sirin Software 2018, ‘The ARM Processors: A, R, and M Categories and Their Specifics’, *Technology*, weblog post, 05 Apr 2018, viewed 30 Apr 2022, < https://sirinsoftware.com/blog/the-arm-processor-a-r-and-m-categories-and-their-specifics/>

Slesareva, N., Bruhn, A. and Weickert, J., 2005, August. Optic flow goes stereo: A variational method for estimating discontinuity-preserving dense disparity maps. In Joint Pattern Recognition Symposium (pp. 33-40). Springer, Berlin, Heidelberg.

Sun, C 1997, December. A fast stereo matching method. In Digital Image Computing: Techniques and Applications (pp. 95-100). Auckland, New Zealand: Massey University.

Sutton, D and Green, R., 2010, November. Evaluation of real time stereo vision system using web cameras. In *2010 25th International Conference of Image and Vision Computing New Zealand* (pp. 1-10). IEEE.

Tensor Flow 2022, TensorFlow Lite for Microcontrollers, *For Mobile and Edge*, viewed 30 Apr 2022, <https://www.tensorflow.org/lite/microcontrollers>

Yoo, H. & Chankov, S., 2018. Drone-Delivery Using Autonomous Mobility: An Innovative Approach to Future Last-mile Delivery Problems, s.l.: IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2018, pp. 1216-1220, doi: 10.1109/IEEM.2018.8607829..

Zbontar, J. and LeCun, Y., 2016. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.*, *17*(1), pp.2287-2318.

Zhang, J., Campbell, J.F., Sweeney II, D.C. and Hupman, A.C., 2021. Energy consumption models for delivery drones: A comparison and assessment. Transportation Research Part D: Transport and Environment, 90, p.102668.

Zhang, Z., 2000. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(11), pp.1330-1334.

Zou, L. and Li, Y., 2010, November. A method of stereo vision matching based on OpenCV. In 2010 International Conference on Audio, Language and Image Processing (pp. 185-190). IEEE.

Appendix A – Project Specification

ENG4111/4112 Research Project

Project Specification

For: Christopher Bourke

Title: Stereovision Autonomous Drop Zone Assessment and UAS Delivery Modification

Major: Mechatronic Engineering

Supervisors: Dr. Tobias Low

Sirigalpatabandige Ruveen Perera

Enrollment: ENG4111 – EXT S1, 2021

ENG4112 – EXT S2, 2021

Project Aim: To develop and implement a medium range stereo vision-based distance measuring system on a UAS platform to assess a drop zone for payload delivery.

Programme: Version 1, 17th March 2021

Conduct initial research on UAS machine vision applications. Explore UAS delivery methods and the legislation surrounding autonomy in UAS applications.

Review methods of calculating depth from stereo vision including the factors most affecting the accuracy and resolution of the output. Investigate differing algorithms for stereo vision.

Conceptualise the components of a suitable integrated system including UAS platform, machine vision components, companion computer, ground control, store delivery and testing systems.

Select hardware and a suitable software development environment. The requirements for necessary capability and costs to inform selection.

Develop a machine vision algorithm for optimized disparity mapping.

Construct an initial prototype to facilitate data collection. Including UAS control programs and fail-safes.

Refine stereo vision matching algorithms and post process data output for optimum use in a UAS platform.

Integrate and deploy the prototype and algorithms at a suitable location and record data for evaluation.

Process and evaluate experimental data.

If time and resources permit:

Refine detection and data processing algorithms, depending on what is achieved earlier. Look at differing hazards (water, object avoidance)

Integrate further assessment algorithms, feature based objects identification (human), water detection, investigate rudimentary object avoidance.

Project Resources

Camera system

  Raspberry Pi HQ camera module (2)

    SONY IMX477

16mm 10MP telephoto lens (2)

Companion computer

Primary

  NVIDIA Jetson Nano Developer Kit-B01

Secondary

  Raspberry Pi compute 4 IO module

  Raspberry Pi compute 4 module

Tertiary

Arducam Stereo Camarray Hat

  Raspberry Pi 4

Stereo Camera Gimbal

  3D printed as required.

UAS platform

  S500 quadrotor

Software

Python – general programming

Raspberry pi OS – companion computer

OpenCV – Machine vision

MATLAB – Machine vision

Creo – 3D printing and drafting

Ardupilot – UAS control, testing and monitoring UAS

Dronekit – Programming mission and behaviour

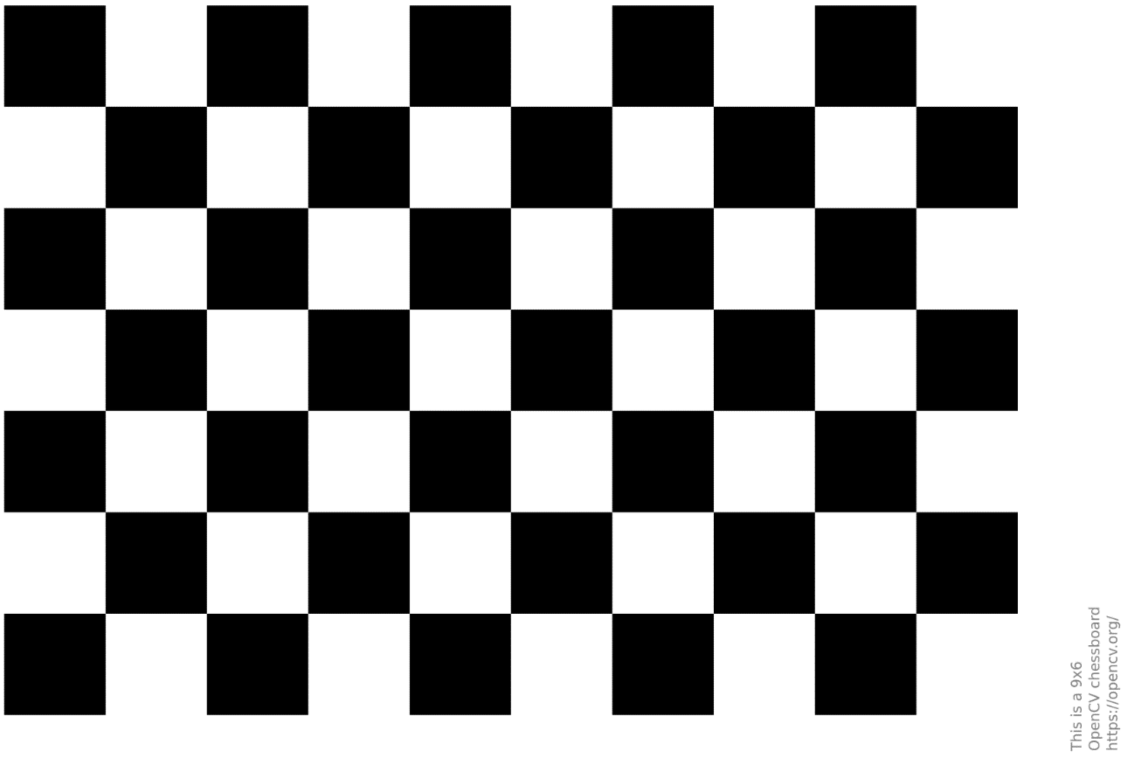
YOLO (object detection post primary research) – Object detection, machine vision

Testing equipment

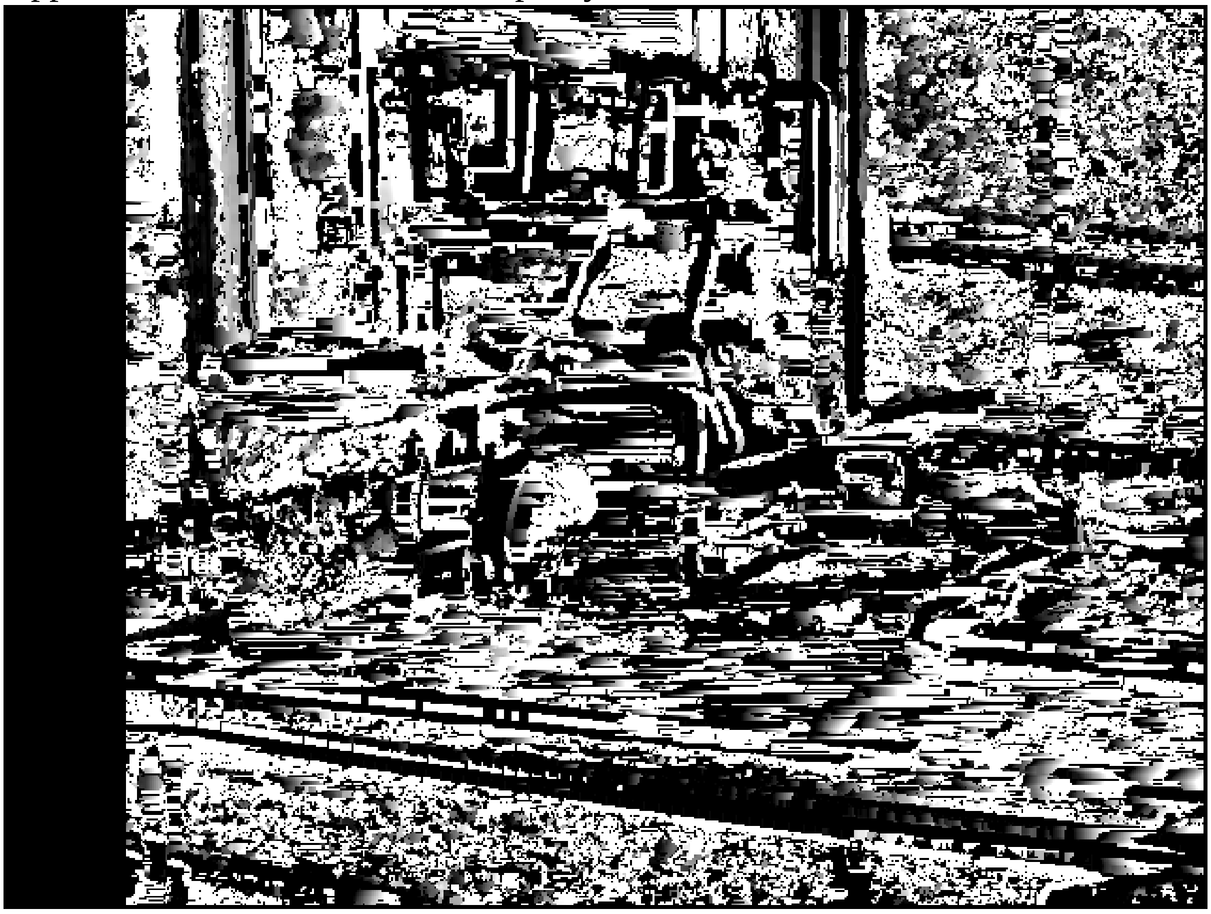
Ground control stations

Video recording equipment (ground and UAS based)

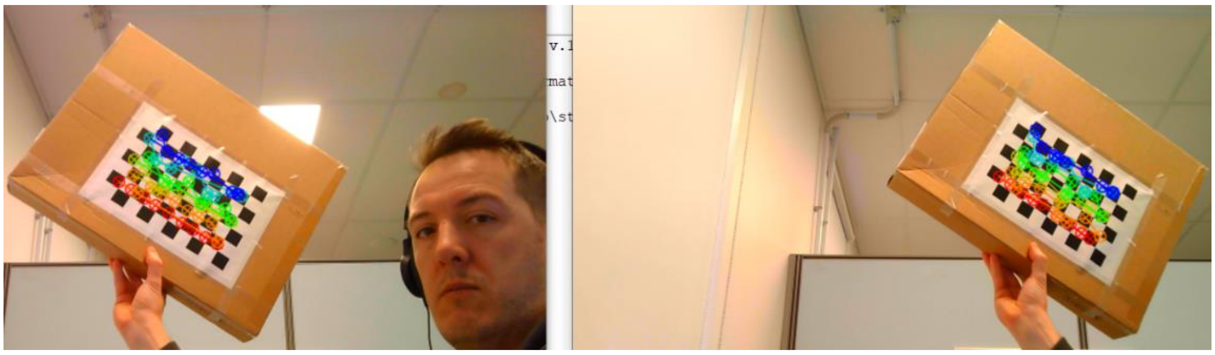
Appendix B - OpenCV Calibration Image

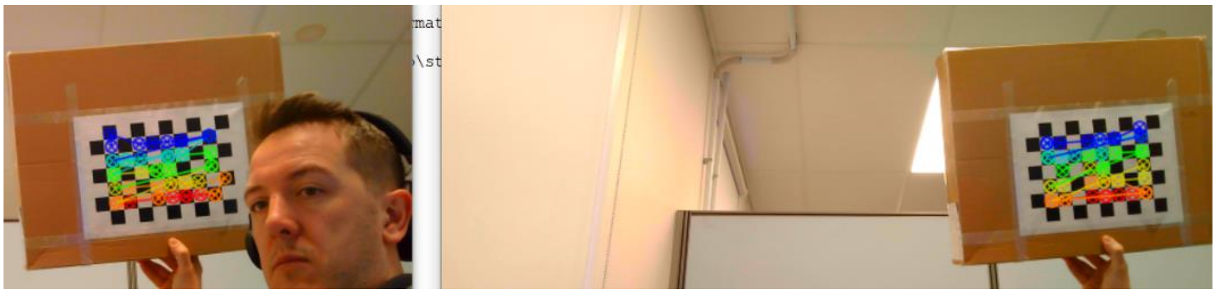


Appendix C - MATLAB Poor Disparity

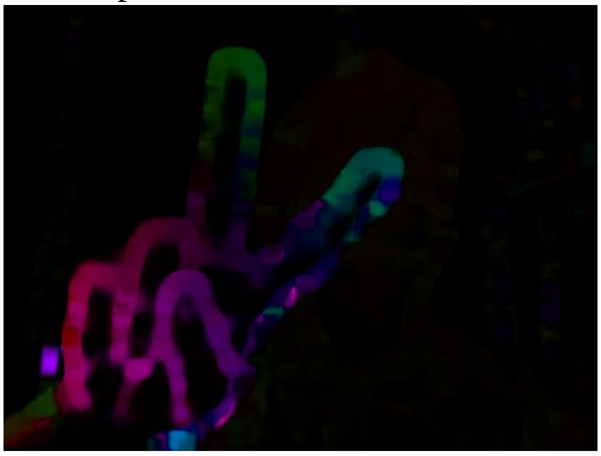


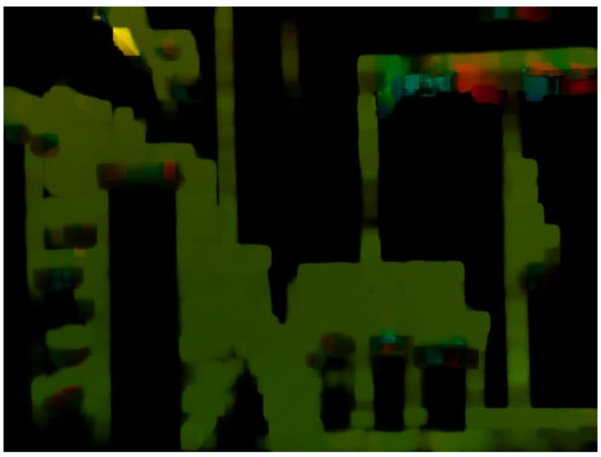
Appendix D - Poor Chessboard Capture



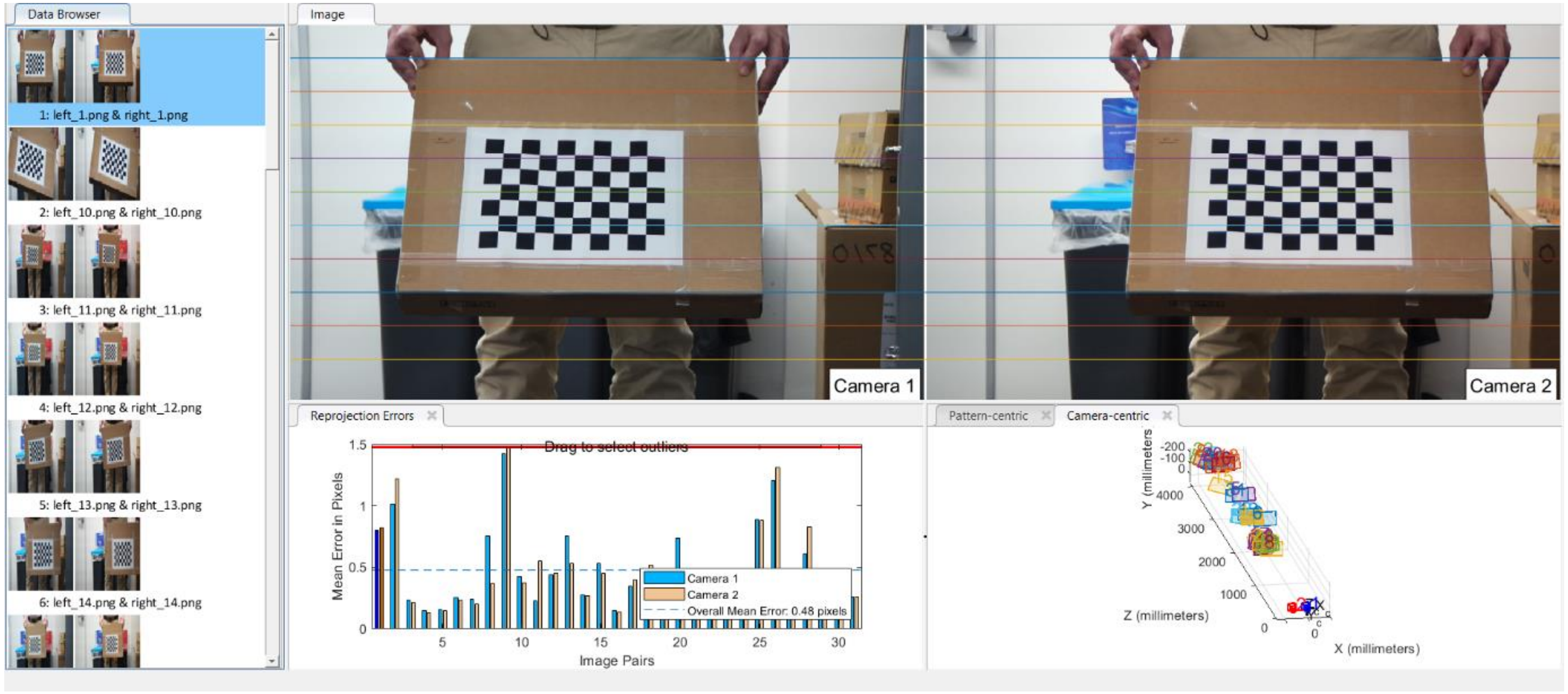


Appendix E - Optical Flow Dense Farnebeck Implementation



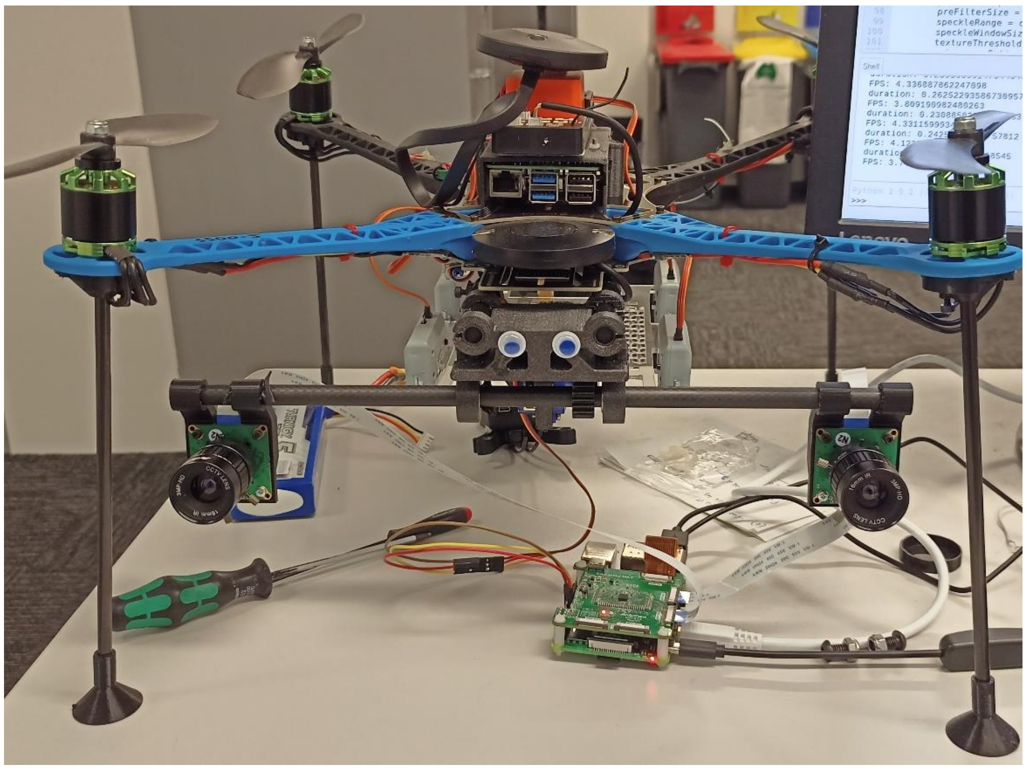


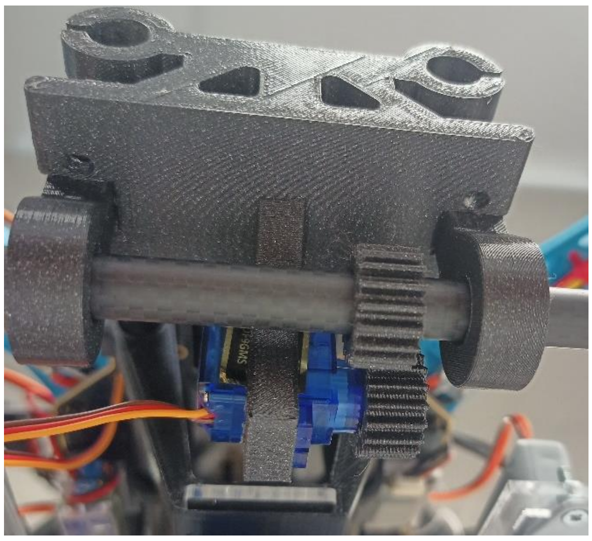
Appendix F – MATLAB Calibration and Rectification with Error



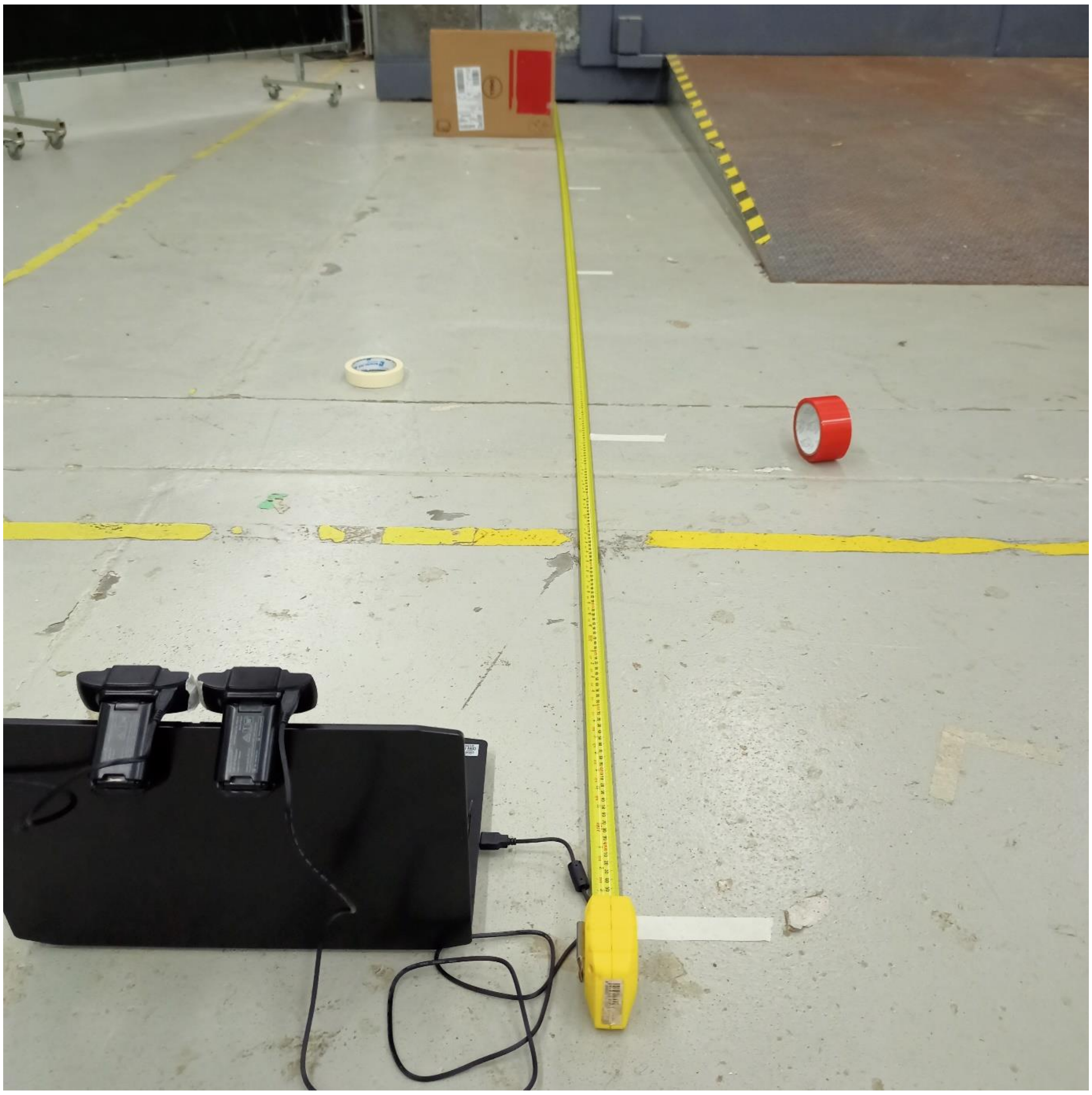
Data Browser
1: left_1.png & right_1.png
2: left_10.png & right_10.png
3: left_11.png & right_11.png
4: left_12.png & right_12.png
5: left_13.png & right_13.png
6: left_14.png & right_14.png
Image
Camera 1
Camera 2
Reprojection Errors
Drag to select outliers
Mean Error in Pixels
Image Pairs
Camera 1
Camera 2
Overall Mean Error: 0.48 pixels
Pattern-centric
Camera-centric
Y (millimeters)
Z (millimeters)
X (millimeters)

Appendix G – Stereo UAS Build and Gimbal Control

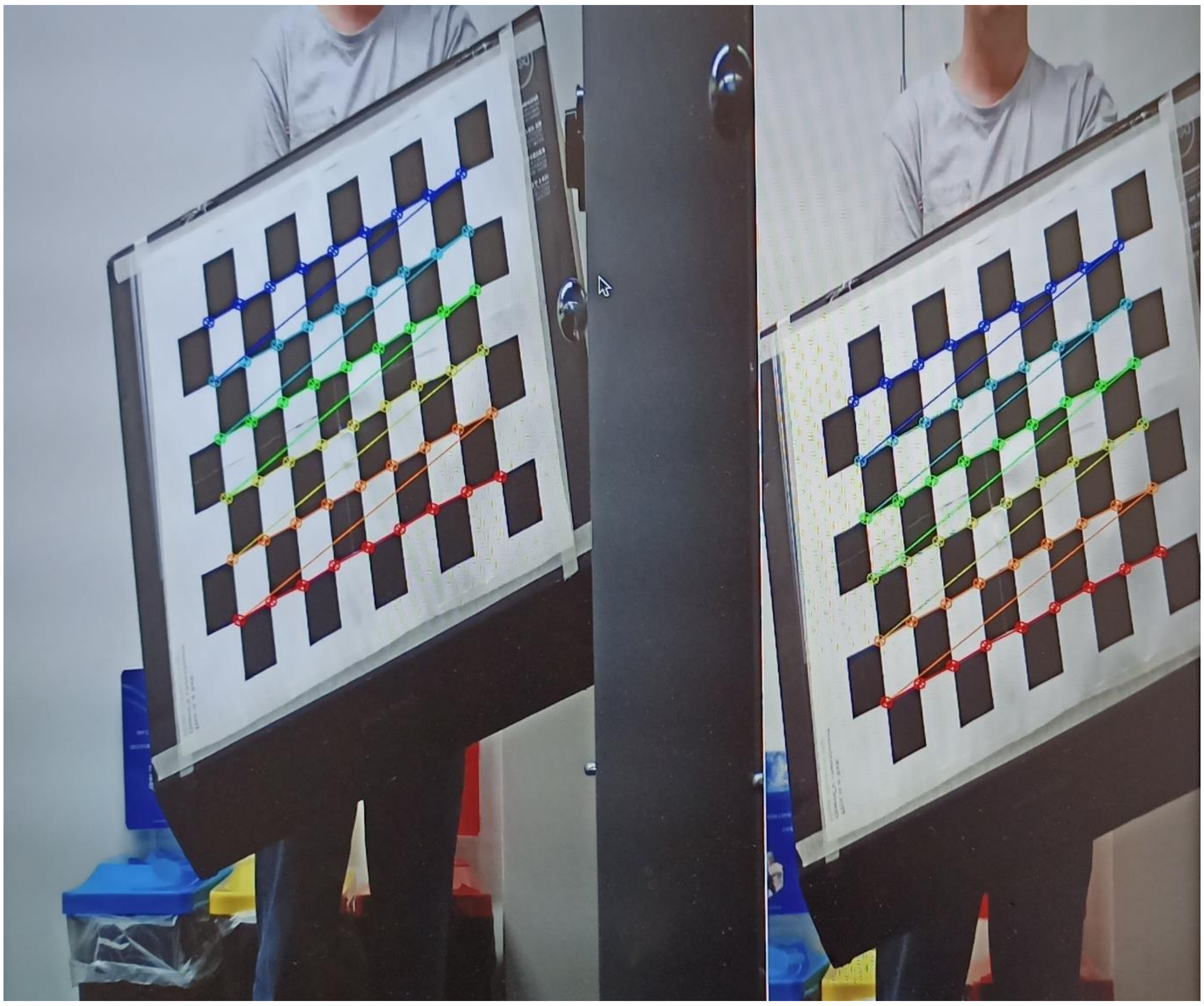




Appendix H - Distance Measurement Setup



Appendix I - Calibration Chessboard Capture A1



Appendix I - Stereo Calibration Code

```
import cv2 as cv
import numpy as np

cap = cv.VideoCapture(0)


##cap.set(3,1080)
##cap.set(4,720)

ret, frame1 = cap.read()
prvs = cv.cvtColor(frame1,cv.COLOR_BGR2GRAY)
hsv = np.zeros_like(frame1)
hsv[...,1] = 255
while True:
    ret, frame2 = cap.read()
    new = cv.cvtColor(frame2, cv.COLOR_BGR2GRAY)
    flow = cv.calcOpticalFlowFarneback(prvs,new, None, 0.5, 3, 15, 3 ,5, 1.2, 0)
    mag, ang = cv.cartToPolar(flow[...,0], flow[...,1])
    hsv[...,0] = ang*180/np.pi/2
    hsv[...,2] = cv.normalize(mag,None,0,255,cv.NORM_MINMAX)
    bgr = cv.cvtColor(hsv,cv.COLOR_HSV2BGR)

#Create and ROI and draw it on the reference
#
#    windowSize = [200,200] #window size
#    bgrSize = bgr.shape
#crop original to new shape
#    bgr = bgr[int((bgrSize[0]/2)-(windowSize[0]/2)):int((bgrSize[0]/2)+(windowSize[0]/2)),int((bgrSize[1]/2)-(windowSize[1]/2)):int((bgrSize[1]/2)+(windowSize[1]/2))]
#    cv.rectangle(frame2, (int((bgrSize[1]/2)-(windowSize[1]/2)), int((bgrSize[1]/2)-(windowSize[1]/2))), (int((bgrSize[0]/2)+(windowSize[0]/2)), int((bgrSize[0]/2)+(windowSize[0]/2))) , (0,0,255), 5)


    cv.imshow('reference', frame2)
    cv.imshow('frame2',bgr)
    k = cv.waitKey(30) & 0xff
    if k == 27:
        break
    elif k == ord('s'):
        cv.imwrite('opticalflowb.png',frame2)
        cv.imwrite('opticalflowhsv.png',bgr)
    prvs = new


cap.release()
cv.destroyAllWindows()
```

Appendix J - Stereo Pyramid Test

```
import cv2 as cv
import numpy as np

#ladder specific details
f = 1733.68 #pixels
b = 0.0022113  #m

###########MIDDLEBURY BENCHMARK TEST ###############
ladL = cv.imread('ladder1_l.png')
ladR = cv.imread('ladder1_r.png')

artL = cv.imread('artroom2_l.png')
artR = cv.imread('artroom2_r.png')

penL = cv.imread('pendulum1_l.png')
penR = cv.imread('pendulum1_r.png')

#this is the exit for the window
def nothing(x):
    pass

def draw_circle(event, x, y, flags, param):
    global mouseX, mouseY
    global f, b
    if event == cv.EVENT_FLAG_LBUTTON:
        cv.circle(disparityNorm,(x,y),5,(0,0,255),2)
        mouseX, mouseY = x,y
        print(mouseX,mouseY)
        disparity = disparityNorm[mouseY,mouseX]
        print('disparity: ' + str(disparity))

        #this is already done
        #f_pixel = (width * 0.5) / np.tan(fov * 0.5 * np.pi/180)

        depth = (b*f)/disparity
        print('depth: ' + str(depth))

#create a window
cv.namedWindow('stereo refine',cv.WINDOW_NORMAL)
cv.resizeWindow('stereo refine', 600,600)

#create a slider (parameter, window, min, max, escape function)
cv.createTrackbar('minDisparity','stereo refine',15,100,nothing)
cv.createTrackbar('numDisparities','stereo refine',3,50,nothing)
cv.createTrackbar('blockSize','stereo refine',7,50,nothing)
cv.createTrackbar('P1','stereo refine',500,500,nothing)
cv.createTrackbar('P2','stereo refine',4000,4000,nothing)
cv.createTrackbar('disp12MaxDiff','stereo refine',0,25,nothing)
cv.createTrackbar('preFilterCap','stereo refine',0,62,nothing)
cv.createTrackbar('uniquenessRatio','stereo refine',0,50,nothing)
cv.createTrackbar('speckleWindowSize','stereo refine',0,400,nothing)
cv.createTrackbar('speckleRange','stereo refine',0,200,nothing)
```

```
window_size = 3
min_disp = 16
num_disp = 112-min_disp

stereo = cv.StereoSGBM_create(min_disp,
                              numDisparities = num_disp,
                              blockSize = 16,
                              P1 = 8*3*window_size**2,
                              P2 = 32*3*window_size**2,
                              disp12MaxDiff = 1,
                              speckleWindowSize = 100,
                              speckleRange = 32
                              )

while True:

    #select the image from the benchmarks
    imgL = ladL
    imgR = ladR

##    #pyramid down
    imgLH = cv.pyrDown(imgL)
    imgLH = cv.resize(imgLH,(480,640))
    imgRH = cv.pyrDown(imgR)
    imgRH = cv.resize(imgRH,(480,640))

    imgLQ = cv.pyrDown(imgLH)
    imgLQ = cv.resize(imgLQ,(480,640))
    imgRQ = cv.pyrDown(imgRH)
    imgRQ = cv.resize(imgRQ,(480,640))

    imgLE = cv.pyrDown(imgLQ)
    imgLE = cv.resize(imgLE,(480,640))
    imgRE = cv.pyrDown(imgRQ)
    imgRE = cv.resize(imgRE,(480,640))

    imgRGray = cv.cvtColor(imgR,cv.COLOR_BGR2GRAY)
    imgLGray = cv.cvtColor(imgL,cv.COLOR_BGR2GRAY)
    imgRGrayH = cv.cvtColor(imgRH,cv.COLOR_BGR2GRAY)
    imgLGrayH = cv.cvtColor(imgLH,cv.COLOR_BGR2GRAY)
    imgRGrayQ = cv.cvtColor(imgRQ,cv.COLOR_BGR2GRAY)
    imgLGrayQ = cv.cvtColor(imgLQ,cv.COLOR_BGR2GRAY)
    imgRGrayE = cv.cvtColor(imgRE,cv.COLOR_BGR2GRAY)
    imgLGrayE = cv.cvtColor(imgLE,cv.COLOR_BGR2GRAY)

    #get tuning values
    minDisparity = cv.getTrackbarPos('minDisparity','stereo refine')
    numDisparities = cv.getTrackbarPos('numDisparities','stereo refine')*16
    blockSize = cv.getTrackbarPos('blockSize','stereo refine')*2 + 5
    P1 = cv.getTrackbarPos('P1','stereo refine')
    P2 = cv.getTrackbarPos('P2','stereo refine')
    disp12MaxDiff = cv.getTrackbarPos('disp12MaxDiff','stereo refine')
    preFilterCap = cv.getTrackbarPos('preFilterCap','stereo refine')
    uniquenessRatio = cv.getTrackbarPos('uniquenessRatio','stereo refine')
    speckleWindowSize = cv.getTrackbarPos('speckleWindowSize','stereo refine')*2
    speckleRange = cv.getTrackbarPos('speckleRange','stereo refine')

    #set tuning values
    stereo.setMinDisparity(minDisparity)
    stereo.setNumDisparities(numDisparities)
    stereo.setBlockSize(blockSize)
    stereo.setP1(P1)
    stereo.setP2(P2)
    stereo.setDisp12MaxDiff(disp12MaxDiff)
    stereo.setPreFilterCap(preFilterCap)
    stereo.setUniquenessRatio(uniquenessRatio)
    stereo.setSpeckleWindowSize(speckleWindowSize)
    stereo.setSpeckleRange(speckleRange)


######SETUP A REGION OF INTEREST#######
    #I could make this an inverse function of the distance to the target

    picRes = np.shape(imgLGray)
    height = picRes[0]
    width = picRes[1]

    #ROIres = (640,480)
    #distanceFunction = 1/distanceToTarget
###since the resolution is so large use a smaller ROI###
    #imgLGray = imgLGray[int(width/2 - 480/2):int(width/2 + 480/2), int(height/2 - 640/2): int(height/2
+ 640/2)]
    #imgRGray = imgRGray[int(width/2 - 480/2):int(width/2 + 480/2), int(height/2 - 640/2): int(height/2
+ 640/2)]
##    imgLGray = imgLGray[:int(height/2), :int(width/2)]
##    imgRGray = imgRGray[:int(height/2), :int(width/2)]
##    imgLGrayH = imgLGrayH[:int((height/2)/2), :int((width/2)/2)]
##    imgRGrayH = imgRGrayH[:int((height/2)/2), :int((width/2)/2)]

    cv.imshow('imgLGray',imgLGray)
##    cv.imshow('imgLGray half',imgLGrayH)
######add a ROI square on the original image######
    #make a 640x480 rectangle centred in the image
##    rectStart = ((int(width/2 - 480/2)),(int(height/2 - 640/2)))
##    rectEnd = ((int(width/2 + 480/2)),(int(height/2 + 640/2)))
##
##    imgL = cv.rectangle(imgL, rectStart, rectEnd, (0,0,255),2)

    disparity = stereo.compute(imgLGray,imgRGray)
    disparityH = stereo.compute(imgLGrayH,imgRGrayH)
    cv.imshow('disparity half',disparityH)
    cv.imshow('disparity',disparity)
    disparityQ = stereo.compute(imgLGrayQ,imgRGrayQ)
    disparityE = stereo.compute(imgLGrayE,imgRGrayE)

#set the type as a float 32
    disparityFloat = disparity.astype(np.float32)/16.0
    disparityFloatH = disparityH.astype(np.float32)/16.0
    disparityFloatQ = disparityQ.astype(np.float32)/16.0
    disparityFloatE = disparityE.astype(np.float32)/16.0

    disparityNorm = (disparityFloat - min_disp)/num_disp
    disparityNormH = (disparityFloatH - min_disp)/num_disp
    disparityNormQ = (disparityFloatQ - min_disp)/num_disp
    disparityNormE = (disparityFloatE - min_disp)/num_disp


    disparityNorm = cv.resize(disparityNorm, (480,640))
    cv.imshow('disparityNorm',disparityNorm)

##    disparityNorm = disparityNorm[200:400, 200:400]
##    closest = np.max(disparityNorm)
##    nearPt = (f*b)/closest
##    print(str(nearPt))
##
    cv.imshow('disparity ROI',disparityNorm)
    cv.imshow('disparityNorm Half',disparityNormH)
    cv.imshow('disparityNorm Quarter',disparityNormQ)
    cv.imshow('disparityNorm Eighth',disparityNormE)
    #cv.imshow('disparity Float', disparityFloat)
    #cv.imshow('disparity', disparity)
    imgL = cv.resize(imgL, (480,640))
    cv.imshow('left', imgL)
    #cv.imshow('left rectified', leftRec)
    #cv.imshow('right rectified', rightRec)
    #cv.applyColorMap(depth, depth, cv.COLORMAP_JET)

#WEIGHTED BLEND
    combo = cv.addWeighted(disparityNorm,1.0,disparityNormH,1.0,0.0)
    combo = cv.addWeighted(combo,1.0,disparityNormQ,1.0,0.0)
    combo = cv.addWeighted(combo,1.0,disparityNormE,1.0,0.0)

    cv.imshow('combo', combo)

    cv.imwrite('dispity_combo.png',combo)
    cv.imwrite('disparityNorm.png',disparityNorm)
#depth is failing becuase of zero division
    #cv.imshow('depth',depth)

    cv.setMouseCallback('disparityNorm',draw_circle)



    if cv.waitKey(1) == 27:
        break

cv.destroyAllWindows()
```

Appendix K - Stereo Calibrator Program

```
import numpy as np
import cv2 as cv
import glob



################ FIND CHESSBOARD CORNERS - OBJECT POINTS AND IMAGE POINTS
#############################

chessboardSize = (9,6)
frameSize = (640, 480)

# termination criteria
criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)

objp = np.zeros((chessboardSize[0] * chessboardSize[1], 3), np.float32)
objp[:,:2] = np.mgrid[0:chessboardSize[0],0:chessboardSize[1]].T.reshape(-1,2)

objpoints = []
imgpointsL = []
imgpointsR = []


###ensure the images are added as pairs
imagesLeft = glob.glob('images/stereoLeft/*.png')
imagesRight = glob.glob('images/stereoRight/*.png')

for imgLeft, imgRight in zip(imagesLeft, imagesRight):

    imgL = cv.imread(imgLeft)
    imgR = cv.imread(imgRight)
    grayL = cv.cvtColor(imgL, cv.COLOR_BGR2GRAY)
    grayR = cv.cvtColor(imgR, cv.COLOR_BGR2GRAY)

    # Find the chess board corners
    retL, cornersL = cv.findChessboardCorners(grayL, chessboardSize, None)
    retR, cornersR = cv.findChessboardCorners(grayR, chessboardSize, None)

    #add points
    if retL and retR == True:

        objpoints.append(objp)

        cornersL = cv.cornerSubPix(grayL, cornersL, (11,11), (-1,-1), criteria)
        imgpointsL.append(cornersL)

        cornersR = cv.cornerSubPix(grayR, cornersR, (11,11), (-1,-1), criteria)
        imgpointsR.append(cornersR)

        # Show captured corners
        cv.drawChessboardCorners(imgL, chessboardSize, cornersL, retL)
##        imgL = cv.resize(imgL,(640,480))
        cv.imshow('img left', imgL)
        cv.drawChessboardCorners(imgR, chessboardSize, cornersR, retR)
##        imgR = cv.resize(imgR,(640,480))
        cv.imshow('img right', imgR)
        cv.waitKey(500)


cv.destroyAllWindows()

############## CALIBRATION #######################################################

retL, cameraMatrixL, distL, rvecsL, tvecsL = cv.calibrateCamera(objpoints, imgpointsL, frameSize, None, None)
heightL, widthL, channelsL = imgL.shape
newCameraMatrixL, roi_L = cv.getOptimalNewCameraMatrix(cameraMatrixL, distL, (widthL, heightL), 1, (widthL, heightL))

###reprojection errors#####
mean_error = 0
for i in range(len(objpoints)):
    imgpointsLP, _ = cv.projectPoints(objpoints[i], rvecsL[i], tvecsL[i], cameraMatrixL, distL)
    error = cv.norm(imgpointsL[i], imgpointsLP, cv.NORM_L2)/len(imgpointsLP)
    mean_error += error
print( "Left total error: {}".format(mean_error/len(objpoints)))

retR, cameraMatrixR, distR, rvecsR, tvecsR = cv.calibrateCamera(objpoints, imgpointsR, frameSize, None, None)
heightR, widthR, channelsR = imgR.shape
newCameraMatrixR, roi_R = cv.getOptimalNewCameraMatrix(cameraMatrixR, distR, (widthR, heightR), 1, (widthR, heightR))

mean_error = 0
for i in range(len(objpoints)):
    imgpointsRP, _ = cv.projectPoints(objpoints[i], rvecsR[i], tvecsR[i], cameraMatrixR, distR)
    error = cv.norm(imgpointsR[i], imgpointsRP, cv.NORM_L2)/len(imgpointsRP)
    mean_error += error
print( "Right total error: {}".format(mean_error/len(objpoints)))


########## Stereo Vision Calibration ##############################################

flags = 0
flags |= cv.CALIB_FIX_INTRINSIC

criteria_stereo= (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)

retStereo, newCameraMatrixL, distL, newCameraMatrixR, distR, rot, trans, essentialMatrix, fundamentalMatrix = cv.stereoCalibrate(objpoints, imgpointsL, imgpointsR, newCameraMatrixL, distL, newCameraMatrixR, distR, grayL.shape[::-1], criteria_stereo, flags)


########## Stereo Rectification #################################################

rectifyScale= 1
```

```
rectL, rectR, projMatrixL, projMatrixR, Q, roi_L, roi_R= cv.stereoRectify(newCameraMatrixL, distL, newCameraMatrixR, distR, grayL.shape[::-1], rot, trans, rectifyScale,(0,0))

stereoMapL = cv.initUndistortRectifyMap(newCameraMatrixL, distL, rectL, projMatrixL, grayL.shape[::-1], cv.CV_16SC2)
stereoMapR = cv.initUndistortRectifyMap(newCameraMatrixR, distR, rectR, projMatrixR, grayR.shape[::-1], cv.CV_16SC2)

print("Writing parameters")
cv_file = cv.FileStorage('stereoMap.xml', cv.FILE_STORAGE_WRITE)

cv_file.write('stereoMapL_x',stereoMapL[0])
cv_file.write('stereoMapL_y',stereoMapL[1])
cv_file.write('stereoMapR_x',stereoMapR[0])
cv_file.write('stereoMapR_y',stereoMapR[1])

cv_file.release()

print("done")
```

Appendix L - Stereo Semi Global Block Matching Distance Measurement (USB)

```
import sys
import cv2 as cv
import numpy as np
import time
import imutils
from matplotlib import pyplot as plt

# Function for stereo vision and depth estimation
import calibration

import time

#ladder specific details
fov = 56.6 #degrees horizontal
b = 0.29  #m

def nothing(x):
    pass

def draw_circle(event, x, y, flags, param):
    global mouseX, mouseY
    global fov, b, width
    if event == cv.EVENT_FLAG_LBUTTON:
        cv.circle(disparityNorm,(x,y),5,(0,0,255),2)
        mouseX, mouseY = x,y
        print(mouseX,mouseY)
        disparity = disparityNorm[mouseY,mouseX]
        print('disparity: ' + str(disparity))

        f_pixel = (width * 0.5) / np.tan(fov * 0.5 * np.pi/180)

        depth = (b*f_pixel)/disparity
        print('depth: ' + str(depth))

# Open both cameras
cap_right = cv.VideoCapture(2, cv.CAP_DSHOW)
cap_left =  cv.VideoCapture(1, cv.CAP_DSHOW)

###resolution change####
##cap_right.set(3,1920)
##cap_right.set(4,1080)
##cap_left.set(3,1920)
##cap_left.set(4,1080)


##################refinement#########################################################

cv.namedWindow('stereo refine',cv.WINDOW_NORMAL)
cv.resizeWindow('stereo refine', 600,600)
```

```
#create a slider (parameter, window, min, max, escape function)
cv.createTrackbar('minDisparity','stereo refine',0,500,nothing)
cv.createTrackbar('numDisparities','stereo refine',100,500,nothing)
cv.createTrackbar('blockSize','stereo refine',9,1920,nothing)
cv.createTrackbar('P1','stereo refine',500,500,nothing)
cv.createTrackbar('P2','stereo refine',4000,4000,nothing)
cv.createTrackbar('disp12MaxDiff','stereo refine',25,25,nothing)
cv.createTrackbar('preFilterCap','stereo refine',0,62,nothing)
cv.createTrackbar('uniquenessRatio','stereo refine',13,100,nothing)
cv.createTrackbar('speckleWindowSize','stereo refine',400,400,nothing)
cv.createTrackbar('speckleRange','stereo refine',200,200,nothing)

###do something with these##
##window_size = 3
##min_disp = 16
##num_disp = 112-min_disp

stereo = cv.StereoSGBM_create()

while True:

    #maybe use grab() and retrieve() here?
##    cap_right.grab()
##    cap_left.grab()
##
##    retL,imgL = cap_right.retrieve()
##    retR,imgR = cap_left.retrieve()

    retL,imgL = cap_left.read()
    retR,imgR = cap_right.read()

    if retL and retR:
        ##CHANGE##
        imgRGray = cv.cvtColor(imgL,cv.COLOR_BGR2GRAY)
        imgLGray = cv.cvtColor(imgR,cv.COLOR_BGR2GRAY)

##################                                          CALIBRATION
#########################################################

        frame_right, frame_left = calibration.undistortRectify(imgRGray, imgLGray)

##################################################################################
######


##        cv.imshow('left rec', frame_left)
##        cv.imshow('right rec', frame_right)



    #get tuning values
        minDisparity = cv.getTrackbarPos('minDisparity','stereo refine')
        numDisparities = cv.getTrackbarPos('numDisparities','stereo refine')
        blockSize = cv.getTrackbarPos('blockSize','stereo refine')
        P1 = cv.getTrackbarPos('P1','stereo refine')
        P2 = cv.getTrackbarPos('P2','stereo refine')
```

```
	disp12MaxDiff = cv.getTrackbarPos('disp12MaxDiff','stereo refine')
	preFilterCap = cv.getTrackbarPos('preFilterCap','stereo refine')
	uniquenessRatio = cv.getTrackbarPos('uniquenessRatio','stereo refine')
	speckleWindowSize = cv.getTrackbarPos('speckleWindowSize','stereo refine')
	speckleRange = cv.getTrackbarPos('speckleRange','stereo refine')


##
	#set tuning values
	stereo.setMinDisparity(minDisparity)
	stereo.setNumDisparities(numDisparities)
	stereo.setBlockSize(blockSize)
	stereo.setP1(P1)
	stereo.setP2(P2)
	stereo.setDisp12MaxDiff(disp12MaxDiff)
	stereo.setPreFilterCap(preFilterCap)
	stereo.setUniquenessRatio(uniquenessRatio)
	stereo.setSpeckleWindowSize(speckleWindowSize)
	stereo.setSpeckleRange(speckleRange)


	disparity = stereo.compute(frame_right,frame_left)

#set the type as a float 32
	disparityFloat = disparity.astype(np.float32)/16.0

	min_disp = minDisparity
	num_disp = numDisparities

	disparityNorm = (disparityFloat - min_disp)/num_disp

#get a distance calculation f(mm)*b(m) = focal length and baseline


	disparityNorm = cv.resize(disparityNorm, (640,480))


	cv.imshow('disparityNorm',disparityNorm)
	#cv.imshow('disparity Float', disparityFloat)
	#cv.imshow('disparity', disparity)
	imgL = cv.resize(imgL, (640,480))
	cv.imshow('left', imgL)
	cv.imshow('left rectified', frame_left)
	#cv.imshow('right rectified', rightRec)
	#cv.applyColorMap(depth, depth, cv.COLORMAP_JET)

#depth is failing becuase of zero division
	#cv.imshow('depth',depth)
	dim = disparityNorm.shape
	width = dim[1]

	cv.setMouseCallback('disparityNorm',draw_circle)

	key = cv.waitKey(1)
	if key == ord('q'):
```

```
            break

    else:
        camL = cv.VideoCapture(1)
        camR = cv.VideoCapture(2)


cap_right.release()
cap_left.release()

cv.destroyAllWindows()
```

Appendix M - Calibration Image Capture (USB)

```
import cv2
import time

capL = cv2.VideoCapture(2)
capR = cv2.VideoCapture(0)

######resolution changes######
##cap.set(3,1920)
##cap.set(4,1080)
##cap2.set(3,1920)
##cap2.set(4,1080)

num = 0
countdown = 5

while capL.isOpened():

    time.sleep(1)

##    succes1, img = cap.read()
##    succes2, img2 = cap2.read()

    capL.grab()
    capR.grab()

    retL, imgL = capL.retrieve()
    retR, imgR = capR.retrieve()


    k = cv2.waitKey(5)

    if k == 27:
        break
##    elif k == ord('s'): # wait for 's' key to save and exit
##        cv2.imwrite('images/stereoLeft/imageL' + str(num) + '.png', img)
##        cv2.imwrite('images/stereoright/imageR' + str(num) + '.png', img2)
##        print("images saved!")
##        num += 1

    imgL = cv2.resize(imgL,(640,480))
    imgR = cv2.resize(imgR,(640,480))

##    img = cv2.putText(img, str(countdown),(10,75),cv2.FONT_HERSHEY_SIMPLEX, 3, (0,255,0),
5)
##    cv2.imshow('Img 1',img)
##    cv2.imshow('Img 2',img2)
    countdown = countdown - 1
    if countdown <= 0:
        cv2.imwrite('images/stereoLeft/imageL' + str(num) + '.png', imgL)
        cv2.imwrite('images/stereoright/imageR' + str(num) + '.png', imgR)
        print("images saved!")
        num += 1
        countdown = 5
    img = cv2.putText(imgL, str(countdown),(10,75),cv2.FONT_HERSHEY_SIMPLEX, 3, (0,255,0),
5)
    cv2.imshow('Img 1',imgL)
    cv2.imshow('Img 2',imgR)


capL.release()
capR.release()
cv2.destroyAllWindows()
```

Appendix N – Continually Calculated Release Point Algorithm Example

```
"""
this is a CCRP draft
should take altitude, speed (vertical and horizontal), the drag and
reynolds number etc.
"""

import math
import random


"""
#pull the sensor data
Vehicle.location.global_frame #WGS84 coordinates and alt (MSL)
Vehicle.location.global_relative_frame #WGS84 plus altitude from a
point
Vehicle.velocity #three components [vx, vy, vz]
Vehicle.airspeed #m/s double
Vehicle.heading #int 0-360
Vehicle.attitude #p,y,r

"""
#dummy data



data = [random.randrange(1,30) for i in range(20)]

print(data)



#vertical

h = -30 #m altitude
uv = 0 #Vehicle.velocity[1] # m/s vertical speed
g = -9.81


disc = uv * uv - 4 *0.5*g* -h

sqrtval = math.sqrt(abs(disc))



    # checking condition for discriminant

if discri > 0:

    time = ((-uv + sqrtval)/(2 * 0.5*g))

    time2 = ((-uv - sqrtval)/(2 * 0.5*g))

    if time2 > 0:
        time = time2
    print('time: ' + str(time))

elif discri == 0:
    print(-uv / (2 * 0.5*g))


#horizontal
#these details are for a tennis ball

ux = 10 #Vehicle.velocity[0]#horizontal speed m/s
mass = 0.0577 #kg
Cd = 0.15 #for the streamlined (3d) body
rho = 1.2041 #kg/m^3 density of air

A = math.pi*0.067**2

dragF = Cd*rho*ux**2*A/2

ax = dragF / mass

for ux in data:
    dist = ux*time + 0.5*ax*time**2
    print('dist: ' + str(dist))

#decision to release

'''
while the heading is on target:
      if stereo depth == gps dist:
            cruise release
      elseif stereo depth < gps dist:
            hover release
'''
```

Appendix O – Calibration Program UAS Implementation

```
import cv2 as cv
import numpy as np
import glob

criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30,
0.001) #after 30 iterations or the specified accuracy has been met

count = 0

#create blank object point arrays (width*height,colour channel)
objp = np.zeros((9*6,3), np.float32)
objp[:,:2] = np.mgrid[0:9,0:6].T.reshape(-1,2)

objpoints = []
imgpointsL = []
imgpointsR = []

#getting files from a folder
images = [image for image in
glob.glob('/home/pi/Desktop/project/good_stuff/calibration_photos/*.png
')] #my chessboard split photos


#make sure the files load as actual pairs
images.sort()


for image in images:

    image = cv.imread(image)
    #this is (height, width, channels)
    shape = image.shape
    width = int(shape[1]/2)
    height = shape[0]

    il = image[0:height,0:width]
    ir = image[0:height, width:2*width]


    grayL = cv.cvtColor(il, cv.COLOR_BGR2GRAY)
    grayR = cv.cvtColor(ir, cv.COLOR_BGR2GRAY)

    retL, cornersL = cv.findChessboardCorners(grayL, (9,6), None)
    retR, cornersR = cv.findChessboardCorners(grayR, (9,6), None)

    if retL and retR == True:
        count = count + 1

        objpoints.append(objp)

        cornersL = cv.cornerSubPix(grayL, cornersL, (11,11), (-1,-1),
criteria)
        imgpointsL.append(cornersL)

        cornersR = cv.cornerSubPix(grayR, cornersR, (11,11), (-1,-1),
criteria)
        imgpointsR.append(cornersR)
        cv.drawChessboardCorners(il, (9,6), cornersL, retL)
#         cv.resize(il, (640,480))
        cv.imshow('imgL', il)

        cv.drawChessboardCorners(ir, (9,6), cornersR, retR)
#         cv.resize(ir,(640,480))
        cv.imshow('imgR', ir)

##assess any innaccurate chessboard captures
        cv.waitKey(500)

cv.destroyAllWindows()

######make corrections given chessboard detections############

retL, mtxL, distL, rvecsL, tvecsL = cv.calibrateCamera(objpoints,
imgpointsL, grayL.shape[::-1], None ,None) #the image size is (w,h) and
shape gives (h,w)
heightL, widthL, channelsL = il.shape
newCameraMtxL, roiL = cv.getOptimalNewCameraMatrix(mtxL, distL,
(widthL, heightL), 1, (widthL, heightL))

######reprojection error#######
mean_error = 0
for i in range(len(objpoints)):
    imgpointsLP, _ = cv.projectPoints(objpoints[i], rvecsL[i],
tvecsL[i], mtxL, distL)
    error = cv.norm(imgpointsL[i], imgpointsLP,
cv.NORM_L2)/len(imgpointsLP)
    mean_error += error
print("Left total error: {}".format(mean_error/len(objpoints)))



retR, mtxR, distR, rvecsR, tvecsR = cv.calibrateCamera(objpoints,
imgpointsR, grayR.shape[::-1], None ,None)
heightR, widthR, channelsR = ir.shape
newCameraMtxR, roiR = cv.getOptimalNewCameraMatrix(mtxR, distR,
(widthR, heightR), 1, (widthR, heightR))

mean_error = 0
for i in range(len(objpoints)):
    imgpointsRP, _ = cv.projectPoints(objpoints[i], rvecsR[i],
tvecsR[i], mtxR, distR)
    error = cv.norm(imgpointsR[i], imgpointsRP,
cv.NORM_L2)/len(imgpointsRP)
    mean_error += error
print("Right total error: {}".format(mean_error/len(objpoints)))



flags = 0
flags |= cv.CALIB_FIX_INTRINSIC

criteria_stereo = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER,
30, 0.001)
```

```
retS, newCameraMtxL, distL, newCameraMtxR, distR, rot, trans,
essentialMtx, fundamentalMtx = cv.stereoCalibrate(objpoints,
imgpointsL, imgpointsR, newCameraMtxL, distL, newCameraMtxR, distR,
grayL.shape[::-1], criteria_stereo, flags)

rectifyScale = 1
rectL, rectR, projMtxL, projMtxR, Q, roiL, roiR =
cv.stereoRectify(newCameraMtxL, distL, newCameraMtxR, distR,
grayL.shape[::-1], rot, trans, rectifyScale,(0,0))

stereoMapL = cv.initUndistortRectifyMap(newCameraMtxL, distL, rectL,
projMtxL, grayL.shape[::-1], cv.CV_16SC2)
stereoMapR = cv.initUndistortRectifyMap(newCameraMtxR, distR, rectR,
projMtxR, grayR.shape[::-1], cv.CV_16SC2)

print(str(count) + " images found")
print("writing params")

cv_file = cv.FileStorage('stereoMap.xml', cv.FILE_STORAGE_WRITE)

cv_file.write('stereoMapL_x', stereoMapL[0])
cv_file.write('stereoMapL_y', stereoMapL[1])
cv_file.write('stereoMapR_x', stereoMapR[0])
cv_file.write('stereoMapR_y', stereoMapR[1])

cv_file.release()

print('done')
```

Appendix P – Stereo Program UAS Implementation

```
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2 as cv


#######setup#########
#call camera
camera = PiCamera()
camera.resolution = (1920,720)
camera.vflip = True
camera.framerate = 32

fov = 24
b = 0.29

rawCapture = PiRGBArray(camera, size =(1920,720))

#let the camera wake up
time.sleep(0.5)


#read cal data from previous
#got to read the stereo)rectify_maps.xml
#the same as the other ones
cv_file = cv.FileStorage("stereoMap.xml", cv.FILE_STORAGE_READ)

stereoMapL_x = cv_file.getNode("stereoMapL_x").mat()
stereoMapL_y = cv_file.getNode("stereoMapL_y").mat()
stereoMapR_x = cv_file.getNode("stereoMapR_x").mat()
stereoMapR_y = cv_file.getNode("stereoMapR_y").mat()

#this is the exit for the window
def nothing(x):
    pass

#######splitter#########
def splitter(img):

    shape = img.shape
    width = int(shape[1]/2)
    height = shape[0]

    il = img[0:height,0:width]
    ir = img[0:height, width:2*width]

    return il,ir

def draw_circle(event, x, y, flags, param):
    global mouseX, mouseY
    global fov, b, width
    if event == cv.EVENT_FLAG_LBUTTON:
        cv.circle(disparityNorm,(x,y),5,(255,0,0),2)
        mouseX, mouseY = x,y
        print(mouseX,mouseY)
        disparity = disparityNorm[mouseY,mouseX]
        print('disparity: ' + str(disparity))

        f_pixel = (width * 0.5) / np.tan(fov * 0.5 * np.pi/180)

        depth = (b*f_pixel)/disparity
        print('depth: ' + str(depth))

# #######Stereo##########
# #create a window
cv.namedWindow('stereo refine',cv.WINDOW_NORMAL)
cv.resizeWindow('stereo refine', 200,200)
#
# #create a slider (parameter, window, min, max, escape function)
cv.createTrackbar('blockSize','stereo refine',10,100,nothing)
cv.createTrackbar('disp12MaxDiff','stereo refine',0,100,nothing)
cv.createTrackbar('minDisparity','stereo refine',23,480,nothing)
cv.createTrackbar('numDisparities','stereo refine',3,50,nothing)
cv.createTrackbar('preFilterCap','stereo refine',62,62,nothing)#63 max
+ 1
cv.createTrackbar('preFilterSize','stereo refine',250,250,nothing)#255
max + 1
#preFilterType = 'XSobel (default), 'NormalizedResponse'
#ROI1 = [0,0,100,100]
#ROI2 = [0,0,100,100]
cv.createTrackbar('speckleRange','stereo refine',200,200,nothing)
cv.createTrackbar('speckleWindowSize','stereo refine',400,400,nothing)
cv.createTrackbar('textureThreshold','stereo refine',100,100,nothing)
cv.createTrackbar('uniquenessRatio','stereo refine',10,200,nothing)


stereo = cv.StereoBM_create() #defaults

######capture##########
while True:



    for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):

        start = time.time()

        image = frame.array
#         image = cv.resize(image, (480,640))
#         cv.imshow('image', image)
        imgL,imgR = splitter(image)

#         cv.imshow('left', imgL)
#         cv.imshow('right', imgR)

#########you've got to designate an ROI #####################

    #maybe use undistort here
        leftRec = cv.remap(imgL, stereoMapL_x, stereoMapL_y,
cv.INTER_LANCZOS4, cv.BORDER_CONSTANT,0)
        rightRec = cv.remap(imgR, stereoMapR_x, stereoMapR_y,
cv.INTER_LANCZOS4, cv.BORDER_CONSTANT,0)
```

```
    #get tuning values
        blockSize = cv.getTrackbarPos('blockSize','stereo refine')+5
        disp12MaxDiff = cv.getTrackbarPos('disp12MaxDiff','stereo
refine')
        minDisparity = cv.getTrackbarPos('minDisparity','stereo
refine')
        numDisparities = cv.getTrackbarPos('numDisparities','stereo
refine')*16
        preFilterCap = cv.getTrackbarPos('preFilterCap','stereo
refine')+1
        preFilterSize = cv.getTrackbarPos('preFilterSize','stereo
refine')+5
        speckleRange = cv.getTrackbarPos('speckleRange','stereo
refine')
        speckleWindowSize =
cv.getTrackbarPos('speckleWindowSize','stereo refine')
        textureThreshold = cv.getTrackbarPos('textureThreshold','stereo
refine')
        uniquenessRatio = cv.getTrackbarPos('uniquenessRatio','stereo
refine')


    #set tuning values
        stereo.setBlockSize(int(np.floor(blockSize)//2*2+1))
        stereo.setDisp12MaxDiff(disp12MaxDiff)
        stereo.setMinDisparity(minDisparity)
        stereo.setNumDisparities(numDisparities)
        stereo.setPreFilterCap(preFilterCap)
        stereo.setPreFilterSize(int(np.floor(preFilterSize)//2*2+1))
        stereo.setSpeckleRange(speckleRange)
        stereo.setSpeckleWindowSize(speckleWindowSize)
        stereo.setTextureThreshold(textureThreshold)
        stereo.setUniquenessRatio(uniquenessRatio)

##Greyscale for disparity matching
        leftRec = cv.cvtColor(leftRec,cv.COLOR_BGR2GRAY)
        rightRec = cv.cvtColor(rightRec,cv.COLOR_BGR2GRAY)

        disparity_BM = stereo.compute(rightRec, leftRec)
        disparityFloat = disparity_BM.astype(np.float32)/16.0
        disparityNorm = (disparityFloat - minDisparity)/numDisparities

        dims = disparityNorm.shape
        height = dims[1]
        width = dims[0]

        cv.rectangle(disparityNorm, (int(height/2-200), int(width/2-
200)), (int(height/2+200),int(width/2+200)), (255, 255, 255), 2)
        disparityNorm = cv.resize(disparityNorm,(640,480))
        cv.imshow('disparityNorm',disparityNorm)

        leftRec = cv.resize(leftRec,(640,480))
        cv.imshow('leftRec',leftRec)



        #FPS calculation
        end = time.time()
        duration = end - start
        fps = 1/duration
#         print('FPS:', fps)

#         dim = disparityNorm.shape
#         width = dim[1]
#
#
#         cv.setMouseCallback('disparityNorm',draw_circle)


#         roi = disparityNorm[int(height/2-100):int(width/2-
100),int(height/2+100):int(width/2+100)]
#         cv.imshow("roi",roi)

        max_disp = np.max(roi)
        f_pixel = (width * 0.5) / np.tan(fov * 0.5 * np.pi/180)
        depth = (b*f_pixel)/max_disp
        print(str(depth))


        rawCapture.truncate(0)

        key = cv.waitKey(1)

        if key == ord('q'):
            break

        else:
            camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True)
            rawCapture.truncate(0)


    cv.destroyAllWindows()
    camera.close()
    break
```

Appendix Q – MATLAB ORB Feature Rectification Program

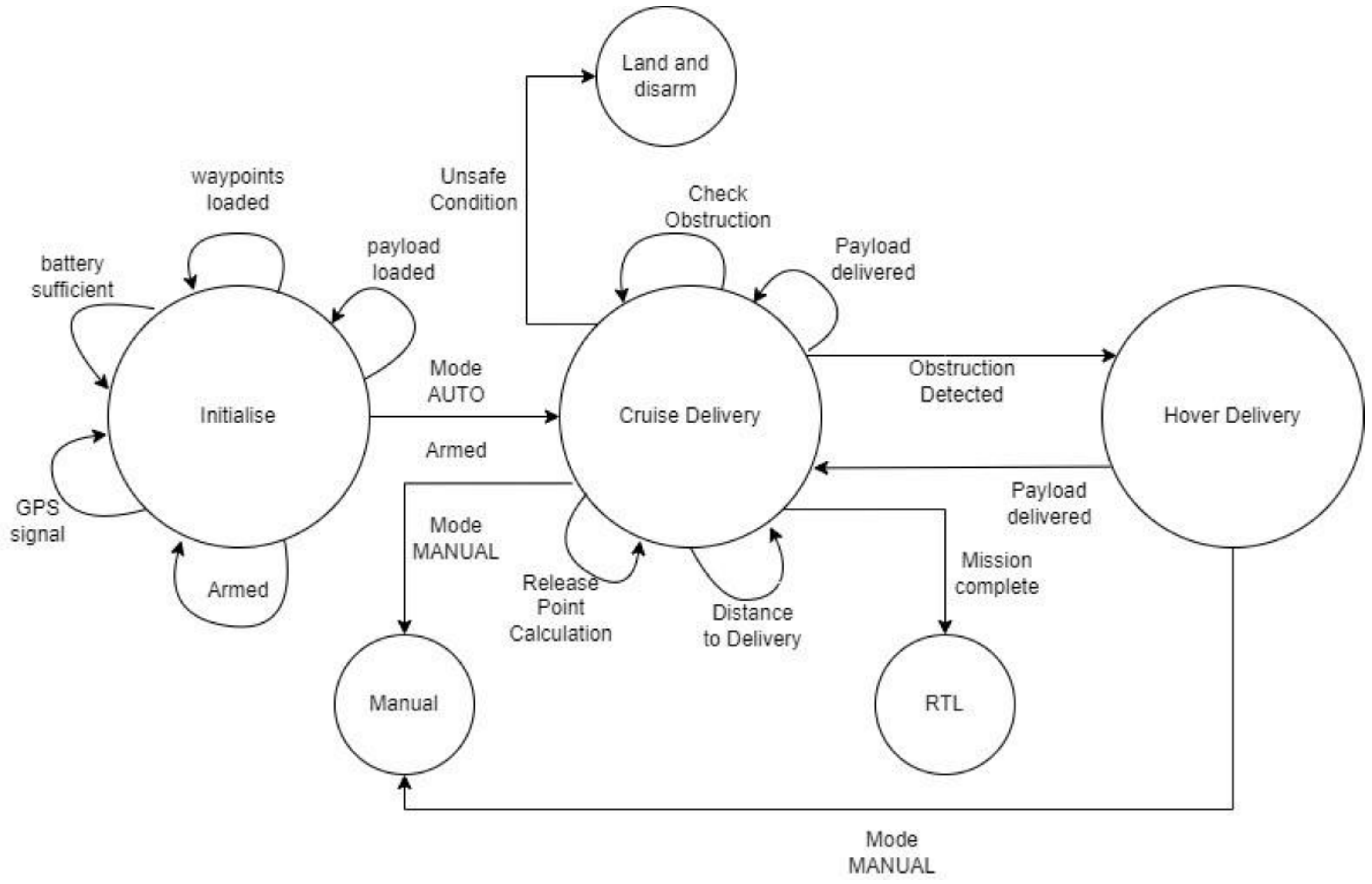
```
I1 = imread('C:\Users\CTBou\Documents\python_trash\left
images\HD\left12', 'png');
I1 = rgb2gray(I1);
%gauss = fspecial('gaussian',5,1);
%lap = [0 -1 0;-1 4 -1; 0 -1 0];
%I1 = conv2(I1, gauss, 'same');
%I1 = conv2(I1, lap,'same');

I2 = imread('C:\Users\CTBou\Documents\python_trash\right
images\HD\right12', 'png');
I2 = rgb2gray(I2);
%I2 = conv2(I2, gauss, 'same');
%I2 = conv2(I2, lap,'same');

figure;
imshowpair(I1,I2,'montage');

title('I1 (left); I2 (right)');
figure;
imshow(stereoAnaglyph(I1,I2));
title('Composite Image (Red - Left Image, Cyan - Right Image');

%%
blobs1 = detectORBFeatures(I1);
blobs2 = detectORBFeatures(I2);

figure;
imshow(I1);
hold on;
plot(selectStrongest(blobs1, 30));
title('30 Strongest ORB Features in I1');

figure;
imshow(I2);
hold on;
plot(selectStrongest(blobs2, 30));
title('30 strongest ORB features in I2');

%%
[features1, validBlobs1] = extractFeatures(I1, blobs1);
[features2, validBlobs2] = extractFeatures(I2, blobs2);

indexPairs = matchFeatures(features1,features2,'Metric','SAD',
'MatchThreshold',10, 'MaxRatio', 0.5);
matchedPoints1 = validBlobs1(indexPairs(:,1),:);
matchedPoints2 = validBlobs2(indexPairs(:,2),:);
figure;
showMatchedFeatures(I1,I2,matchedPoints1,matchedPoints2);

%%
[fMatrix, epipolarInliers, status] =
estimateFundamentalMatrix(matchedPoints1, matchedPoints2, 'Method',
'Norm8Point',...
    'Numtrials', 10000, 'DistanceThreshold', 0.1, 'Confidence', 99.99);
if isEpipoleInImage(fMatrix, size(I1)) || isEpipoleInImage(fMatrix',
size(I2))
```

```
    error(['either not enough matching points were found or the
epipoles are inside the images.']);
end

inlierPoints1 = matchedPoints1(epipolarInliers, :);
inlierPoints2 = matchedPoints2(epipolarInliers, :);

figure;
showMatchedFeatures(I1,I2, inlierPoints1, inlierPoints2);

%%
[t1, t2] = estimateUncalibratedRectification(fMatrix,
inlierPoints1.Location, inlierPoints2.Location, size(I2));
tform1 = projective2d(t1);
tform2 = projective2d(t2);

[I1Rect, I2Rect] = rectifyStereoImages(I1,I2, tform1, tform2);
figure;
imshow(stereoAnaglyph(I1Rect, I2Rect));

%%
disparityRange = [0, 8*16]; %0 to 128
uniquenessRatio = 20; %typically 5-15, 0 to disable
disparityMapI = disparitySGM(I1Rect,I2Rect,'DisparityRange',
disparityRange, 'UniquenessThreshold', uniquenessRatio);
figure;
imshow(disparityMapI, [0,15]);
colormap jet
colorbar
```

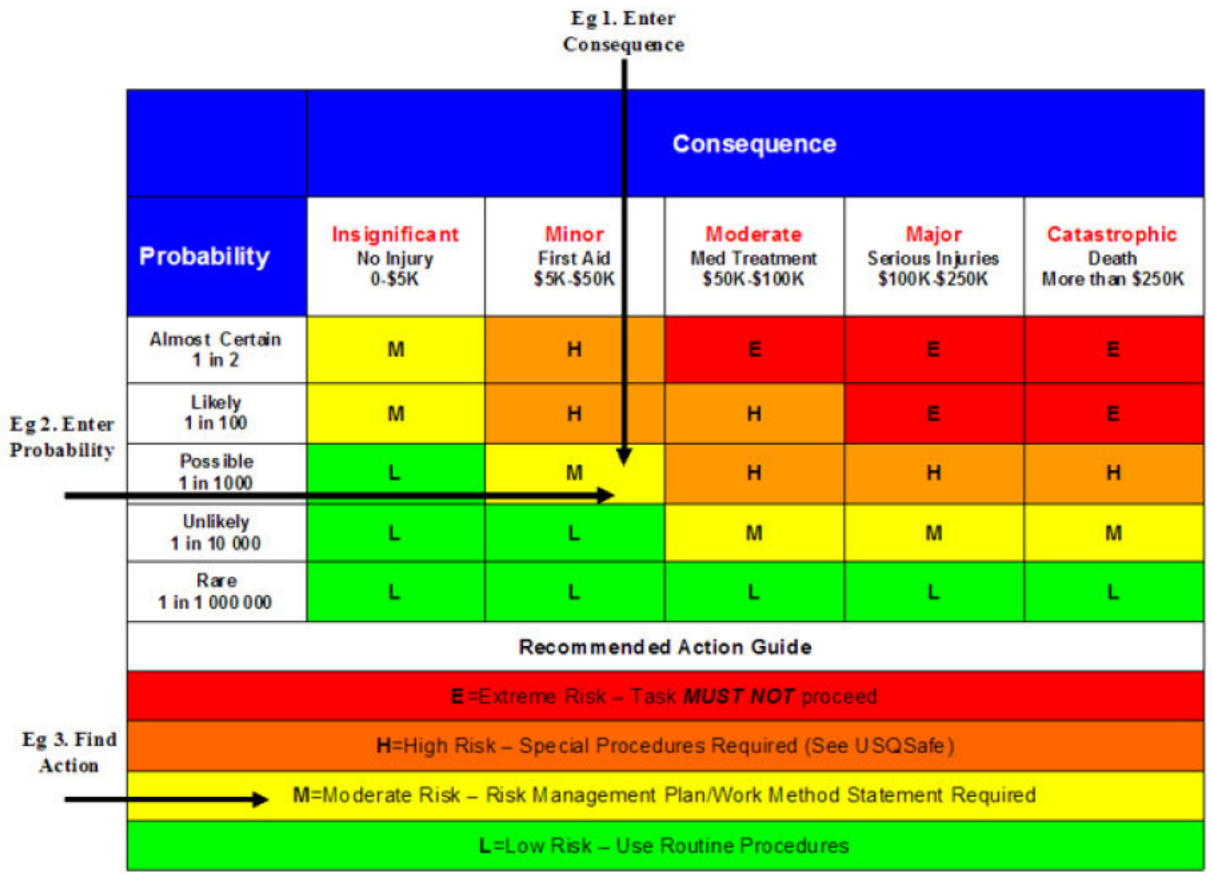
Appendix R – MATLAB Disparity Calcuation Program

```
%load('stereoParams.mat')
I1 = imread('BFS_left','jpg');
I2 = imread('BFS_right','jpg');
%[I1,I2] = rectifyStereoImages(I1,I2,stereoParams);
%I1 = rgb2gray(J1);
%I2 = rgb2gray(J2);
I1 = rgb2gray(I1);
I2 = rgb2gray(I2);
figure;
imshow(stereoAnaglyph(I1,I2));
%figure;
%imshow(stereoAnaglyph(J1,J2));
disparityRange = [0, 16*5];
disparityMapI = disparitySGM(I1,I2,'DisparityRange', disparityRange,
'UniquenessThreshold', 10);
figure;
imshow(disparityMapI, [0,64]);
title('Disparity Map');
colormap jet
colorbar
```

Appendix S - State Machine Diagram Mission Execution



Appendix T – USQ Safety Risk Management System


Eg 1. Enter Consequence
Eg 2. Enter Probability
Eg 3. Find Action

	Consequence				
Probability	**Insignificant** No Injury 0-$5K	**Minor** First Aid $5K-$50K	**Moderate** Med Treatment $50K-$100K	**Major** Serious Injuries $100K-$250K	**Catastrophic** Death More than $250K
Almost Certain 1 in 2	M	H	E	E	E
Likely 1 in 100	M	H	H	E	E
Possible 1 in 1000	L	M	H	H	H
Unlikely 1 in 10 000	L	L	M	M	M
Rare 1 in 1 000 000	L	L	L	L	L

Recommended Action Guide

E=Extreme Risk – Task ***MUST NOT*** proceed

H=High Risk – Special Procedures Required (See USQSafe)

M=Moderate Risk – Risk Management Plan/Work Method Statement Required

L=Low Risk – Use Routine Procedures

Step 1 (cont)	Step 2	Step 2a	Step 2b	Step 3			Step 4				
Hazards: From step 1 or more if identified	***The Risk:*** What can happen if exposed to the hazard without existing controls in place?	***Consequence:*** What is the harm that can be caused by the hazard without existing controls in place?	***Existing Controls:*** What are the existing controls that are already in place?	***Risk Assessment:*** Consequence x Probability = Risk Level			***Additional controls:*** Enter additional controls if required to reduce the risk level	***Risk assessment with additional controls:***			
				Probability	**Risk Level**	**ALARP? Yes/no**		**Consequence**	**Probability**	**Risk Level**	**ALARP? Yes/no**
Example											
Working in temperatures over 35° C	Heat stress/heat stroke/exhaustion leading to serious personal injury/death	catastrophic	Regular breaks, chilled water available, loose clothing, fatigue management policy.	possible	high	No	temporary shade shelters, essential tasks only, close supervision, buddy system	catastrophic	unlikely	mod	Yes
Autonomous UAS operation	UAS impact	Moderate	Enclosed observation demountable building on a restricted access open area designated for UAS testing can protect personnel involved in the area of operation. Building in a geofence in the mission planning to ensure that the UAS breaks out of autonomous operation and moves back into the designated operational area. Manual remote pilot operator on standby to take over if the operation is unexpected or dangerous.	Rare	Low	Yes		Select a consequence	Select a probability	Select a Risk Level	Yes or No
Li-po battery operation	Risk of fire or explosion is mistreated	Moderate	Battery boxes for transport. Fit-for-purpose charging apparatus. Disconnecting batteries until ready for operation	Rare	Low	Yes		Select a consequence	Select a probability	Select a Risk Level	Yes or No

Step 1 (cont)	Step 2	Step 2a	Step 2b	Step 3			Step 4				
Hazards: From step 1 or more if identified	***The Risk:*** What can happen if exposed to the hazard without existing controls in place?	***Consequence:*** What is the harm that can be caused by the hazard without existing controls in place?	***Existing Controls:*** What are the existing controls that are already in place?	***Risk Assessment:*** Consequence x Probability = Risk Level			***Additional controls:*** Enter additional controls if required to reduce the risk level	***Risk assessment with additional controls:***			
				Probability	**Risk Level**	**ALARP? Yes/no**		**Consequence**	**Probability**	**Risk Level**	**ALARP? Yes/no**
Example											
Working in temperatures over 35° C	Heat stress/heat stroke/exhaustion leading to serious personal injury/death	catastrophic	Regular breaks, chilled water available, loose clothing, fatigue management policy.	possible	high	No	temporary shade shelters, essential tasks only, close supervision, buddy system	catastrophic	unlikely	mod	Yes
Unresponsive Flight	Run away UAS	Minor	Automatic geofences. Manual override. Allocated and booked testing area.	Rare	Low	Yes		Select a consequence	Select a probability	Select a Risk Level	Yes or No
Payload realease	Uncommanded release or store impact	Minor	UAS testing (SITL) prior to testing Designated observation area Manual override GPS RTK for accurate positioning of delivery area and start positions Positioning area away from structure and personnel	Unlikely	Low	Yes		Select a consequence	Select a probability	Select a Risk Level	Yes or No
Manual UAV handling	Risk of rotor strike	Moderate	Battery connection as late as possible Manual override and operator in place during setup External arming selection for the UAS to ensure power can	Rare	Low	Yes		Select a consequence	Select a probability	Select a Risk Level	Yes or No

Step 1 (cont)	Step 2	Step 2a	Step 2b	Step 3			Step 4				
Hazards: From step 1 or more if identified	***The Risk:*** What can happen if exposed to the hazard without existing controls in place?	***Consequence:*** What is the harm that can be caused by the hazard without existing controls in place?	***Existing Controls:*** What are the existing controls that are already in place?	***Risk Assessment:*** Consequence x Probability = Risk Level			***Additional controls:*** Enter additional controls if required to reduce the risk level	***Risk assessment with additional controls:***			
				Probability	Risk Level	ALARP? Yes/no		Consequence	Probability	Risk Level	ALARP? Yes/no
Example											
Working in temperatures over 35° C	Heat stress/heat stroke/exhaustion leading to serious personal injury/death	catastrophic	Regular breaks, chilled water available, loose clothing, fatigue management policy.	possible	high	No	temporary shade shelters, essential tasks only, close supervision, buddy system	catastrophic	unlikely	mod	Yes
			be applied without the likelihood of rotor movement.								
Electrical safety when designing UAS	Large capacity batteries and varying loads or votlages constituting a risk to personnel.	Minor	Ensure BECs are appropriate rated for loads Ensure not to overload the system Ensure the battery capacity is appropriate for the motors and UAS weight	Rare	Low	Yes		Select a consequence	Select a probability	Select a Risk Level	Yes or No
Testing outdoors in inclement weather	Excessive heat or cold or environmental risks	Minor	Sunscreen, water and first aid kits available for testing personnel. First aid trained personnel SITL testing prior to live trials	Rare	Low	Yes		Select a consequence	Select a probability	Select a Risk Level	Yes or No
Wildlife interaction	Kangaroos and birdlife at risk of injury or risk on injuring personnel	Minor	Enclosed monitoring area. Vehicles available to clear the area. Reduced testing times after SITL testing	Rare	Low	Yes		Select a consequence	Select a probability	Select a Risk Level	Yes or No

Step 5 - Action Plan (for controls not already in place)

Additional controls:	*Resources:*	*Persons responsible:*	*Proposed implementation date:*
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.

Step 6 - Approval

Drafter's name:	Christopher Bourke	Draft date:	25/05/2022
Drafter's comments:	Interim Risk assessment to be reviewed prior to testing.s		
Approver's name:		Approver's title/position:	
Approver's comments:			
I am satisfied that the risks are as low as reasonably practicable and that the resources required will be provided.			
Approver's signature:		Approval date:	Click here to enter a date.