Development of a Computer Vision Framework for Improved Remotely Piloted Aircraft Operations

by

Brendan Xin-Zhi Ooi

A Thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Aerospace Engineering
Department of Mechanical and Aerospace Engineering

Carleton University
Ottawa, Ontario, Canada
December 2019

The undersigned recommend to
the Faculty of Graduate Studies and Research
acceptance of the Thesis

Development of a Computer Vision Framework for Improved Remotely Piloted Aircraft Operations

Submitted by **Brendan Xin-Zhi Ooi**
in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Aerospace Engineering

_____

Dr. J. Laliberté, Supervisor

_____

Dr. R. Miller, Department Chair

Carleton University

2019

Abstract

This thesis proposes a computer vision framework to enable improved operations of Remotely Piloted Aircraft equipped with onboard image sensors. The main use of payload image sensors is to provide visual imagery data to the system for real-time or post-processing applications; the application of an image quality metric and the ground sampling distance of the image sensor can be used to predict the performance of an image sensor in enabling the image classification task. This information is used to determine the mission-specific operational envelope of the aircraft, to ensure that visual data quality requirements are met. The application of a convolutional neural network for image processing is also presented. Finally, a vision-based positioning system is developed; it achieves an average position estimation difference of 18 cm compared to a commercially available indoor localization system and provides a position update rate at 12 Hz.

Acknowledgements

I would like to take this opportunity to express my gratitude towards the many people who have helped me throughout my journey in the master's program here at Carleton University.

First and foremost, I would like to thank my research supervisor, Dr. Jeremy Laliberté, for being an excellent mentor and for providing me with the privilege to carry out my research under his supervision. He has provided me with a wealth of knowledge in the field of engineering, providing guidance and advice on approaching my research in the field of computer vision that was new to me. Thank you for the many opportunities to attend numerous conferences to network, publish, and present our work on the various projects that I have been a part of. Thank you as well for providing me with the opportunity to further develop my people management skills by allowing me to take the lead on overseeing the various flight activities that our research group conducted.

Thank you to my family members for their continued support and love as I pursued my graduate studies here in Canada. Being away from home is never easy, but you have always supported me through all the decisions and choices I have made for the betterment of my future.

Next, I would like to thank the members of the Carleton UAV research group who I have had the pleasure of working with throughout my time with the group. Thank you for supporting me in my research and assisting in the many flight campaigns that were conducted over the course of the last two years. Special thanks to Loughlin Tuck, Salman Shafi, and Niall McCallum for sharing your knowledge with me in the realm of geophysics

and RPAS operations. Thank you to Mila Kanevsky and Olivia Chamberland for helping with the various flight work needed for my numerous side research projects.

A special thanks to my graduate school family in the MC 3041 office for their continued support and friendship as I worked through my research. Many of you were always willing to take the time to help me, be it for experimental testing or to be a sounding board for my ideas. A special mention goes out to Chris Bassindale, John O'Keefe, and Zachary Copeland for always keeping me going.

I would also like to thank the MAE departmental staff for their help and support throughout my time with Carleton University. Thank you to Vicki Button, Allyson Fitch, Janet Perras, and Dan Premachuk for always getting me room reservations to conduct my experimental testing and to host meetings with visiting guests for my research group. Thank you to Steve Truttman, David Raude, and Stephan Biljan for helping me with numerous testing activities for both my research and my teaching responsibilities. A special thanks to Alex Proctor and the shop team for always being available to assist me in manufacturing parts for experimental testing applications.

Table of Contents

List of Tables

List of Illustrations

List of Appendices

List of Acronyms

Acronyms	Definition
ANN	Artificial Neural Network
API	Application Programming Interface
BVLOS	Beyond Visual Line of Sight
CNN	Convolutional Neural Network
DGPS	Differential Global Positioning System
EO	Electro-Optical
FOV	Field of View
FPV	First-Person View
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GR	Ground Range
GSD	Ground Sampling Distance
IDE	Interactive Development Environment
INS	Inertial Navigation System
IR	Infrared
KCF	Kernalized Correlation Filter
LDM	Laser Distance Meter
OpenCV	Open Source Computer Vision
RGB	Red-Green-Blue

RPA	Remotely Piloted Aircraft
RPAS	Remotely Piloted Aircraft System(s)
SAA	Sense-and-Avoid
SGL	Sanders Geophysics Limited
SfM	Structure from Motion
UAV	Uninhabited Aerial Vehicle
VPS	Vision-based Positioning System
VTOL	Vertical Takeoff and Landing
3DR	3DRobotics

Chapter 1: Introduction

1.1 Thesis Motivation and Overview

The advancement in the miniaturization of sensors and electronic components has led to the rapid development of Remotely Piloted Aircraft Systems (RPAS). To date, many types of RPAS are commercially available for data collection purposes, ranging from the traditional fixed-wing configuration to the more recently developed multi-rotor platforms. Image sensors are one of the most commonly found payloads on Remotely Piloted Aircraft (RPA).

The application of RPA to various industries is heavily influenced by emerging use cases. The use of an image sensor provides an RPAS with a bird's eye view of the mission area and is usually a large component of the data collected during flight. Furthermore, the development and optimization of various computer vision processing algorithms have enabled the use of an image sensor to provide real-time information for aircraft task-automation.

One of the most common forms of localization used for RPA is achieved using Global Navigation Satellite System (GNSS) technology. GNSS technology enables an RPA operator to easily identify the location of their aircraft during flight. Numerous satellite constellations available for commercial use (such as the Global Positioning System (GPS) and the Galileo system) allow for coverage on a global scale. However, there are regions where satellite coverage is degraded or inaccessible, which affects the localization capabilities of an RPA. Additionally, certain tasks such as precision landing or following may require localization accuracy that a GNSS is unable to provide.

Advancements in imaging hardware and software have made it possible to develop a framework for the improved application of image sensors onboard an RPA. From a mission data collection perspective, it is important to ensure that the image sensor is capable of providing the desired level of image quality for post-processing applications; for real-time vision-based control applications, the level of accuracy desired for localization applications and the onboard equipment required for image processing must be harmonized. Some of the key motivations for the presented work are as follows:

- Understanding of relationship between image sensor specifications and the achievable data quality.
- Application of machine learning algorithms for automation of imagery data processing.
- Determination of the feasibility of a vision-based localization system for multi-vehicle RPA operations.

1.2 Problem Statement

This thesis sets out to develop a framework for how the information collected by an image sensor placed onboard an RPA can be fully maximized. From a data analysis perspective, it will provide a high-level outline of how an image sensor can be qualitatively assessed to ensure that it can meet the desired mission data quality requirements. This will provide the RPAS operator with information on the capabilities of the image sensor payload. With that knowledge, it is possible for the operator to further develop the RPA survey mission to maximize the quality of aerial imagery data recorded and minimize the need for repeated flights due to insufficient image quality.

From a localization perspective, a vision system can provide a wide array of information from just one sensor. This can be used to augment existing localization techniques, or to replace them for application-specific situations where traditional methods are insufficient or inoperable. This system should also be able to interface with existing RPAS hardware to enable easy integration and application. For this work, a Pixhawk flight controller is selected.

This thesis will focus on the applications of small RPA, which is defined by regulations as aircraft that weigh between 250 g and 25 kg (Part 107 for the United States (Government Publishing Office, 2016) and CAR 900s for Canada (Government of Canada, 2019)), as these are the most easily accessible platforms for research and industrial applications. Current regulations in many countries permit the use of small RPA with a standardized licensing system, which allows for regulation of the knowledge base needed for safe operation (JARUS, 2019; Library of Congress, 2019).

1.3 Organization of Thesis

Chapter 2 of this thesis presents the results of a literature review performed to determine the current state-of-the-art in the use of computer vision on RPAS. First, a high-level review of computer vision and image processing is discussed. Next, the application of a convolutional neural network (CNN) for image processing is presented, focusing on pre-existing models that are readily available for use. The application of vision systems for various types of RPA task-automation is reviewed and summarized. Finally, the application of small RPA to three specific use cases, which drove the development of the work presented in this thesis is detailed.

The research methodologies adopted for this thesis are presented in Chapter 3. An overview of the proposed computer vision framework is provided, and the uses of a Vision-based Positioning System (VPS) for RPAS localization is presented. The selected RPA platforms are discussed, and the various software and hardware associated with the computer vision framework are detailed.

Chapters 4 through 6 cover the applications of specific portions of the proposed computer vision framework, providing details behind the applied concepts as well as the experimental testing conducted to verify its validity. The results of the tests conducted are presented within each chapter directly. A brief conclusion is also provided at the end of each chapter to summarize the results of the work, and how it can be tied into the image processing framework.

The final chapter of this thesis, Chapter 7, ties all the work presented in this thesis together to provide some concluding remarks regarding the application of the work done towards the proposed computer vision framework for improved RPA operations. A summary of the contributions of the work from this thesis is provided. Recommendations for future work, including additional capabilities that can stem from the presented research, are also presented in this chapter.

Chapter 2: Literature Review and Background

This chapter presents an overview of the existing work that has been done in areas related to the work presented in this thesis; it is broken down into four major sections, and are as follows:

- 2.1 Computer Vision and Image Processing
- 2.2 Convolutional Neural Networks for Image Processing
- 2.3 Application of Vision Systems on Remotely Piloted Aircraft
- 2.4 Application-Specific Related Work

2.1 Computer Vision and Image Processing

Computer vision is the field of image processing that began in the 1960s revolving around the use of an image sensor to provide computers with the ability to "see" its surroundings. This field of research is closely linked to the early days of artificial intelligence, as the output of a vision system would allow a suitably programmed computer to make inferences on the presented image and make choices or predictions based on it. To enable this, researchers developed various computer vision algorithms that focused on the extraction of key features within the image frame. Upon extracting these key features, a computer vision system is also able to track and follow these defined features in between image frames. For the context of the further review of literature for computer vision development in this section, it is limited to the functionalities that are available in the Open Source Computer Vision (OpenCV) image processing library, as it is widely used by industry and research organizations for the development and application of computer vision systems.

2.1.1 Feature Extraction

The process of extracting features from an image is the basis for almost all computer vision processes. Just as a human develops their perception of what an object is based on correlation of different features, the same process must be undertaken by a computer for a vision-based system. Feature extractors are developed using traditional image processing operators to identify key features unique to the objects in the image frame, such as corners, edges, and blobs. Table 2.1 provides an overview of some of the most widely used detection operators used for feature extraction.

Table 2.1: Common detectors used in OpenCV

Detector Name	Detected Feature(s)
Harris Corner Detector (Harris & Stephens, 1988)	Corners
Shi-Tomasi Corner Detector (Shi & Tomasi, 1994)	Corners
Canny Edge Detector (Canny, 1986)	Edges
Laplacian of Gaussian (Lindeberg, 1998)	Blobs

Once the computer has located the key features, they must be correlated to one another by identifying regions in the image frame that may be classified as an object. Noble provides a summary of the various feature detectors and matchers available in the OpenCV image processing library (Noble, 2016). Table 2.2 is a summary of the performance results of the feature detectors that are readily available for use in OpenCV obtained from Noble's work.

Table 2.2: Performance of feature detectors in OpenCV - adapted from (Noble, 2016)

Feature Detector	Features Detected	Run Time (ms)
Scale-Invariant Feature Transform (SIFT)	1368	2093
Speeded-Up Robust Features (SURF)	1907	1539
Binary Robust Invariant Scalable Key-points (BRISK)	1132	151
Oriented FAST and Rotated (ORB)	500	207
KAZE	933	7135
Accelerated KAZE	769	2748

2.1.2 Object Tracking

Aside from detecting features in an image frame, another key operation of interest is the ability to track the identified features between successive image frames. Object tracking algorithms generally perform more quickly than the detector algorithms presented in the previous subsection. Once an object has been identified, the system has access to information on said object's appearance and last known location in the image frame; this information can then be used to predict the position of the object in the following frame. This can be a computationally heavy process, dependent on the number of features to be tracked and the distance travelled in between image frames. Trackers can also return false positive identifications of the tracked object, which is caused by the tracker passing over a region in the image where similar features in both the background and object exists. Some tracking algorithms continue to track targets that undergo significant body rotations, while others can recover a tracked target that is subjected to occlusion. By using an object tracking algorithm, it is also possible to identify unique targets within the image frame, which may be helpful for certain computer vision applications, such as counting unique objects throughout a video. Rosebrock (Rosebrock, 2018) and Mallick (Mallick, 2017)

each provide an overview of the various trackers that are available in OpenCV and review the performance of each.

2.2 Convolutional Neural Networks for Image Processing Applications

With the rapid advancement in the realm of machine learning, the use of Artificial Neural Networks (ANN) to process and manage large data sources has become common place. This is no different with the task of image processing; each image is a considerably large amount of data on its own – this concept is covered further in depth in Chapter 5. This section of the literature review looks to provide a brief summary of the open source CNN models that have been trained for applications towards object identification of common day-to-day objects, such as cars, cats, and cups. Table 2.3 is a summary of the performance of some pre-trained CNN models that operate on spatial exploitation for image processing applications, adapted from the review conducted by Khan et al. (Khan, Sohail, Zahoora, & Qureshi, 2019). The error rates reported are based off datasets that have been used as benchmark sets for CNN development. The depth value provides information on the number of layers that exist in the CNN model architecture; this value represents a summation of the total number of convolution, pooling, and fully-connected layers. A larger value represents a more complex CNN architecture; this requires greater computational power to process the image through the model, but generally provides a lower error rate.

Table 2.3: Performance and setup of existing CNN models - adapted from (Khan, Sohail, Zahoora, & Qureshi, 2019)

Architecture	Error Rate	Depth
LeNet (1998)	MNIST: 0.95	7
AlexNet (2012)	ImageNet: 16.4	8
ZefNet (2014)	ImageNet: 11.7	8
VGG (2014)	ImageNet: 7.3	19
GoogLeNet (2015)	ImageNet: 6.7	22
ResNexT (2017)	ImageNet: 4.4	101

CNNs that operate on spatial exploitation allow the developer to modify many parameters for each of the layers. By modifying the parameters of the different layers present in the CNN, it is possible to improve or degrade the performance of the model. The operational concept of a CNN is discussed in Chapter 5.

2.3 Application of Vision Systems on Remotely Piloted Aircraft

Vision systems have been developed for a variety of applications on RPA. These applications range from basic image processing tasks such as target identification to aircraft flight automation for precision maneuvers.

2.3.1 Target Detection and Tracking

One of the key uses of a vision system is to be able to identify targets and to track them within the image frame. Once an image frame has been processed, it is possible to identify different objects of interest that can then be tracked between frames. This application is useful for RPAS that have a focus on monitoring and surveillance activities.

A cooperative vision-based tracking system can enable an RPA to work with others to track and monitor a ground target (Chichella, Kaminer, Dobrokhodov, & Hovakimyan,

2013). The image sensor provides each individual aircraft with a relative location of the target within the image frame; each aircraft can then relay information regarding the position of the tracked target for flight path coordination to avoid collision. Similarly, the task of detection and tracking can also be used to identify other cooperative RPA; this task becomes more complex as the tracked object can move in 3-dimensional space. One method of identifying and tracking cooperative RPA is the use of adaptive template matching and morphological filtering to identify known targets (Opromolla, Vetrella, Fasano, & Accardo, 2018). Adaptive template matching begins with a known template of the tracked object and is updated between each image frame; this minimizes the chances of the mismatching due to variations in the background. In frames where a match cannot be found, the authors applied a morphological filter leveraging known characteristics of the target (color variation from background) for re-identification purposes. Vetrella et al. presented on the application of a vision-based tracking system coupled with a Differential GPS (DGPS) to improve the navigation and performance of one unit in a multi-vehicle system (Vetrella, Fasano, Accardo, & Moccia, 2016). The primary aircraft was retrofitted with both a vision system and a DGPS for localization applications. By tracking two support aircraft with the vision system and sharing of GPS information, both vision and DGPS systems onboard the primary aircraft are used to gain an improvement in its position and velocity estimation.

Another application of target tracking and following was presented by Peng et al. for a non-cooperative setting (Peng, Shiyu, Lin, & Chen, 2013). For this application, no communication exists between both the tracker and target aircraft; a static feature point on the target is tracked, and the distance between the two are estimated by the vision system.

Furthermore, given the known change in distance between image frames, it is also possible to estimate the relative velocity and acceleration of the target. With this, a control system can be developed to enable an RPA to perform desirable maneuvers based on the position of the target, such as maintaining a relative position or orbiting around it, as presented by Razzanelli et al. (Razzanelli, Innocenti, Pannocchia, & Pollini, 2019). In this work, a model predictive controller leveraging the information provided by a vision system was developed to provide trajectory generation for following and tracking another aircraft.

2.3.2 Guidance and Navigation

In the realm of guidance and navigation, there are multiple methods that have been explored for which a vision system can be used as either a primary or supplementary sensor to enable better performance of the RPA.

The most common application of a computer vision system onboard an RPA is to use it in tandem with other onboard sensors to further enhance its localization capabilities. Zhang et al. (Zhang, Chen, Song, & Xu, 2014) utilizes an image sensor to take periodic images of the ground below the aircraft's flight path. These images are then stitched together using computer vision techniques combined with GPS latitude and longitude information to perform real-time image mapping; when the aircraft encounters a situation where GPS information is degraded or unavailable, the system is able to use the real-time video from the images to correlate its position, together with the onboard inertial navigation system (INS) to localize itself and continue to perform autonomous tasks above the previously mapped region. Another use of the vision system and GPS/INS combination is to enable navigation by identifying known landmarks within the image frame, and storing their approximate GPS coordinates (Luo & Pei, 2007); this can be used with a forward

facing camera, and allows the system to localize itself with respect to the known landmarks that were previously tagged.

Vision systems have also been used onboard RPA to assist in enhancing the attitude estimation capabilities of the system. By utilizing an onboard image sensor and optical flow algorithms, it is possible to better estimate the attitude of an aircraft by correlating the vision information with the GPS/INS sensors (Hosen, Helgesen, Fusini, Fossen, & Johansen, 2016). The development of a stable, nonlinear observer based on the velocity field results obtained via optical flow allows for adjustments to the prediction of states of the RPA, which is then fused with the state estimations from the other systems. This results in a greater state estimator due to input from multiple sources.

Choi, Geeves, Alsalam, and Gonzalez were able to leverage the information provided by a vision system to enable on-board decision making for their RPA (Choi, Geeves, Alsalam, & Gonzalez, 2016). In this work, a decision-making tree was developed and programmed on-board to enable the RPA to perform specific maneuvers along its pre-planned flight path. For the first test, the RPA was able to terminate its existing flight plan and land nearby upon identifying the defined target. In the second test, the RPA was programmed to descend to a lower altitude upon detection of the target by the vision system; once the target was out of the image frame, it returned to its original flight plan and proceeded with its mission. The use of a vision system in partnership with a decision-making algorithm can enable dynamic survey missions, enabling the RPA to perform target-specific maneuvers without affecting the overall mission.

2.3.3 Sense-and-Avoid

Sense-and-Avoid (SAA) is another automation task that will benefit from being run onboard the aircraft. For traditional aviation, the presence of a pilot onboard the aircraft, aided by the various onboard sensors, serves as the onboard SAA system. However, this is not possible for an RPA, as the remote pilot has minimal information on the aircraft's surrounding aside from visual cues from the ground due to current regulations that prohibit the incorporation of a first-person view (FPV) device in SAA systems (Government of Canada, 2019). With the increasing use of RPA by industry and researchers for data collection applications, the development of SAA systems will be paramount to enable safe operations of aircraft in increasingly crowded airspace. The development of a SAA system will also be crucial for future beyond visual line of sight (BVLOS) applications, as the pilot would not be able to perform any of the SAA tasks as they have no visual confirmation of the aircraft.

Chand et al. conducted a comprehensive review of the various SAA technologies that could be integrated onto RPAS (Chand, Mahalakshmi, & Naidu, 2017). The review covered technologies that can be applied to both small- and large-scale RPA; however, as the focus of this thesis relates to small RPA, only results pertaining to these are presented in Table 2.4. Co-operative sensors require other aircraft to be equipped with a similar system to enable detection; non co-operative sensors enable detection of any aircraft that is within the system's detection range.

Table 2.4: Sense-and-Avoid sensors for small RPA - adapted from (Chand, Mahalakshmi, & Naidu, 2017)

Sensor Name	Application	Sensor Type
Automatic Dependent Surveillance and Broadcasting (ADS-B)	Relative position and relative velocity	Co-operative
Laser	Range, azimuth, and elevation	Non co-operative
Electro-optic Sensor	Azimuth and elevation	Non co-operative
Acoustic	Range	Non co-operative

A vision system can provide information on everything that is present in the image frame; this includes targets that are both known and unknown. Using the computer vision techniques presented in Section 2.1, it is possible to identify and track a wide range of targets. Initially, the development of SAA systems was focused on obstacle avoidance, such as static targets or the terrain. One method of avoiding static targets mid-flight is to identify regions within the image frame with the least number of obstacles (Martins, Ramos, Braga, Ribeiro, & Mora-Camino, 2017). The authors demonstrated this concept by navigating an RPA through an open door; by exploiting the difference in light intensity and color gradient of the open door versus the walls, a threshold can be set on the pixel intensity values, forcing each pixel to either be “on” or “off”. Based on this, the system could then identify the region with the greatest number of bright pixels, which would signify the region with the least number of obstacles, and to proceed towards that region. As the aircraft moved, this information is updated, and the aircraft can navigate around static obstacles.

A SAA system is essentially another identify and track problem, like the work reviewed in Section 2.2.1. Bharati et al. discusses the application of a Kernalized

Correlation Filter (KCF) tracker for applications toward a SAA system (Bharati, Wu, Sui, Padgett, & Wang, 2018); Dolph et al. developed a SAA system that isolated the search region of the algorithm to objects above the horizon – this system was specifically designed to assist in preventing mid-air collisions, and was able to ignore potential obstacles that were identified on the ground (Dolph, et al., 2017).

2.4 Application-Specific Related Work

The development of the computer vision framework for improved RPA operations presented in Chapter 3 of this thesis was driven by three emerging use-cases; the following subsections present an overview of these three cases, as well as an overview of related work that has been done towards the application of RPAS for automated data collection.

2.4.1 Geophysical Surveys

Carleton University has partnered with Sanders Geophysics Limited (SGL) on the design and development of RPA for applications towards geophysics surveys for many years. Current manned missions carried out by SGL involve the use of modified small manned aircraft, such as the de Havilland DHC-6 Twin Otter and the AS350 B3 Airbus Helicopter, to carry specialized geophysical survey equipment to collect airborne data (SGL, 2019). Several iterations of fixed-wing RPA (Geosurv II, Corvus) and the associated onboard systems have been developed by Carleton University faculty and students to serve as an RPA platform for such surveys. Verbickas applied a CNN for image processing to enable horizon detection for SAA applications (Verbickas, 2012); Owlia looked at developing maneuvers for obstacle avoidance of RPA operating at low altitudes (Owlia, 2013).

Geophysical surveys rely heavily on the use of magnetometers to collect airborne geophysical data to map geological structures and to locate mineral deposits. These sensors are highly sensitive to the magnetic fields emitted by electronics, which is a big challenge for small RPA, as the sensors would be placed near the onboard electronics due to space constraints. Tuck developed a methodology for characterizing the magnetic interference emitted by RPA and how to compensate for it when interpreting airborne magnetic data (Tuck, 2019).

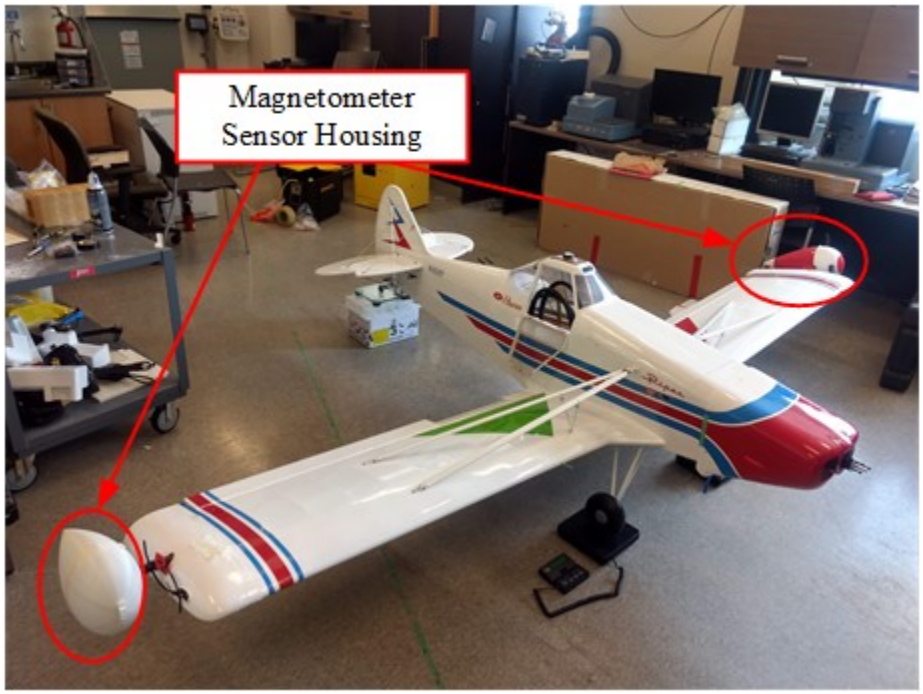

Figure 2.1: Piper Pawnee 1/3rd Scale RPA

The current fixed-wing aircraft under development for use on geophysical surveys at Carleton University is the Pawnee fixed-wing RPA, as depicted in Figure 2.1. This aircraft is an off-the-shelf model that was converted to an electric propulsion system and has been retrofitted with magnetometers for airborne magnetic surveys. The position of the sensors must be known to be able to post-process the collected data to form a magnetic map of the surveyed area. Although a fixed wing platform allows for easy determination

of the relative position of each magnetometer to the other, it is difficult to operate in remote regions due to the unavailability of a runway to enable takeoff and landing. The application of a vision system can enable each aircraft in a fleet of small RPA with vertical takeoff and landing (VTOL) capabilities to accurately determine their position (and by extension, the sensor) in relation to the other operating aircraft. This will enable the use of pre-existing data collection and post-processing methodologies for RPA-based magnetic survey data, allowing for operations in previously inaccessible areas with a fixed-wing aircraft while speeding up the data collection process using multiple sensors spread across the multi-vehicle system.

2.4.2 Infrastructure Inspections

The application of an RPA for infrastructure inspections enables access to hard-to-reach areas, such as building roofs or bridge undercarriages, without putting the operator at risk. These inspections range from visual inspections (Barrile, Candela, Fotia, & Bernardo, 2019; Omar & Nehdi, 2017) to physical testing of the structure (Yamada, Nakao, Hada, & Sawasaki, 2017; Ichikawa, et al., 2017). The image sensor used for the inspection task must have enough resolution to enable the detection of missing bolts and nuts or cracks along the structure. Furthermore, the use of computer vision techniques has enabled the automation of detection of various defects in the structure (Ellenberg, Branco, Krick, Bartoli, & Kontsos, 2015; Jung, Lee, & Kim, 2018).

Early detection of cracks and missing fasteners are paramount to maintaining the structural integrity and safety of the structure. Factors that must be considered for such missions include the operating environment of the RPA, the size of the inspection area, and the proximity of the aircraft to the inspection target. An image sensor with a suitable

resolution must be selected to meet these mission requirements to enable a successful visual inspection.

2.4.3 Wildlife Monitoring

The application of RPA for wildlife monitoring has seen increased use by researchers to assist in conducting a census of the animal population (Hodgson, Baylis, Mott, Herrod, & Clarke, 2016). The task of monitoring wildlife revolves around the minimization of contact between the human observer and the studied species. An RPA can provide an aerial view of the target and its surroundings, allowing researchers to study the animals from a distance while obtaining visual imagery data. The miniaturization of imaging sensors has led to the incorporation of electro-optical (EO) and infrared (IR) sensors onboard RPA to enable detection of wildlife during the day and night (Burke, et al., 2019).

The visual data collected onboard the RPA can be used to enable the counting of different animal species. This aerial view minimizes the need for traditional ground-based image capture sensors that have a limited operational range due to their position on the ground (Gonzalez, et al., 2016). To automate the animal counting process, Kellenberger et al. and Rey et al. leverage the image processing capabilities of a CNN to assist in analyzing the large volume of data collected by an RPA for a given survey area (Kellenberger, Marcos, & Tuia, 2018; Rey, Volpi, Joost, & Tuia, 2017). This application requires an image sensor that can capture the necessary visual details for either a visual observer or a CNN to perform the task of identifying and counting the various animal species contained in the image frame.

Chapter 3: Methodology

The application of a computer vision system on RPAS is a highly customizable process; this is due to the wide array of information that can be derived from an image sensor as well as the large number of image sensors readily available on the market. As such, it is important to ensure that the selected sensor can meet the mission requirements of the RPAS operation. This chapter aims to provide an overview of the research methodologies applied to the work completed in this thesis, as well as the various systems used to enable the proposed computer vision framework for improved RPA operations.

3.1 Computer Vision Framework

To fully maximize the functionality of an image sensor, it is first important to understand the capabilities of such a system. An image sensor can be used to provide one of two types of data sources – still images and video recordings. Currently, it is possible to stream the live video feed from the image sensor across various connections, such as WiFi or via the Internet. Furthermore, the advancement in high speed data transference over these mediums have led to the ability to easily access the increasingly larger video formats in real-time.

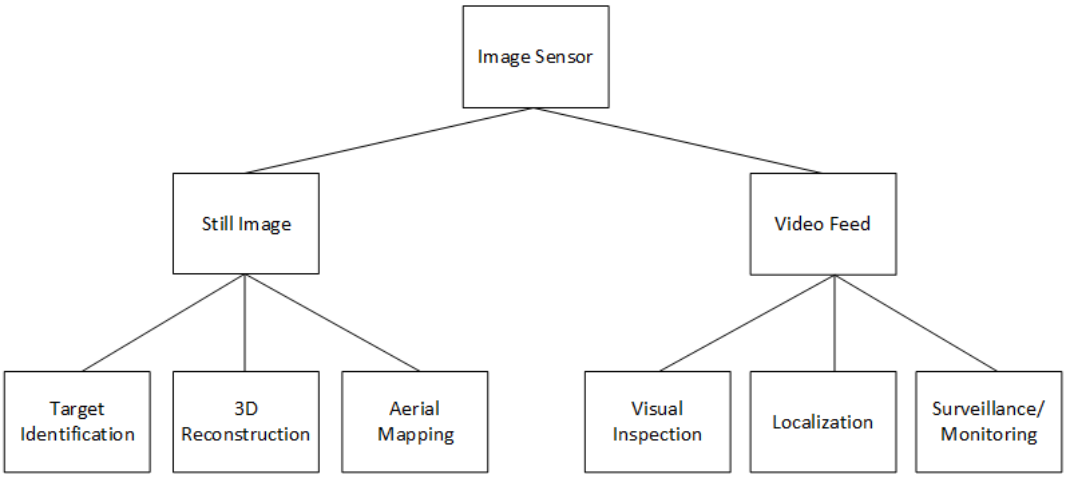

Figure 3.1: Image sensor data collection uses

With the increasing capabilities of image sensors, it is possible to obtain visual imagery data with greater image quality, thus allowing for capturing high levels of detail in the image frame. As with all types of data, the larger the volume of data, the greater the amount of information that can be derived from it; this is no different with an image or video. However, an image or video with a higher resolution does not necessarily mean that the information that it provides is more useful than a lower resolution equivalent.

For small RPA, the applications of a vision system for data collection or real-time controls can be a challenging task. This is due to the significant effect that the weight of an RPA has on its performance and endurance. Higher resolution image sensors result in greater detail captured in the visual information collected; however, this is a double-edged sword as it requires greater computational power to process the image for real-time applications. Additional computing resources would require the need for more hardware to be carried onboard, resulting in a weight penalty that will diminish the performance of the RPA.

To fully utilize the information that can be derived from an image sensor, the processing of imagery data can be broken down into several different steps; Figure 3.2 provides an illustration of the proposed computer vision processing framework. Table 3.1 provides further explanation on each of the four major tasks carried out by the computer vision system. By following this framework, it is possible to identify the key uses of the image sensor onboard the RPA, and it can be developed accordingly. Separating the computer vision framework into different categories enables the vision system to be developed according to the needs of the application, allowing for reduced hardware

requirements for less complex applications; this will assist in addressing the potential weight penalty issue that was previously discussed.

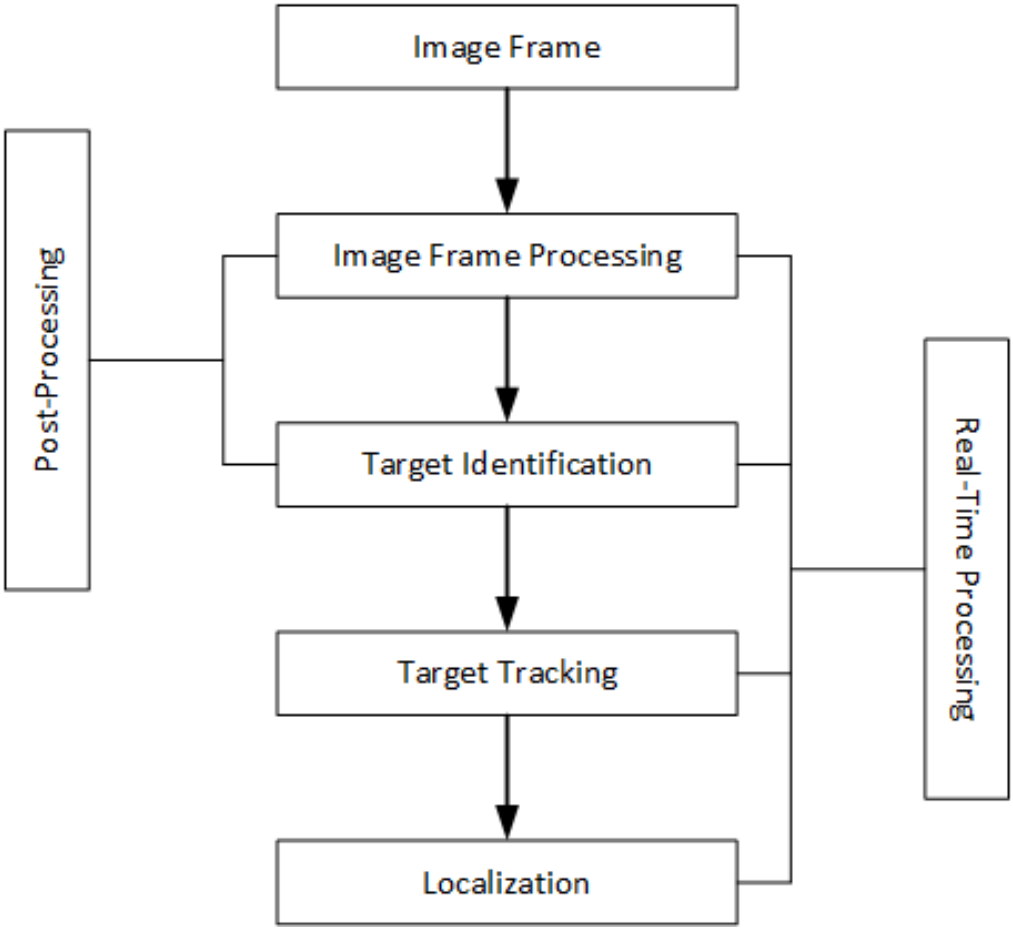

Figure 3.2: Computer vision processing framework

For localization applications, a Vision-based Positioning System (VPS) can be setup utilizing a calibrated image sensor onboard the RPA and a suitably programmed computer system to perform the necessary image processing computation operations. This VPS system should be able to provide information on the surrounding objects to the RPA to enable the determination of its relative position. This information can then enable task-automation applications such as collision avoidance and target tracking.

Table 3.1: Computer vision processing framework tasks

Task	Definition
Image Frame Processing	The first step in the computer vision framework; basic processing occurs in this step, such as edge and corner detection.
Target Identification	Correlation of information found in the first step to begin identifying objects in the image frame; designation of a target can occur in this step if desired.
Target Tracking	Tracking of a defined target in the image frame; various trackers have been developed to enable the tracking of a target across successive image frames.
Localization	Determination of pose information from surrounding objects; can provide information like GNSS and other localization systems for RPA to enable task-automation.

3.2 Remotely Piloted Aircraft Systems (RPAS)

Throughout the course of the work performed for this thesis, several different RPAS were used to conduct the associated flight-testing work. As the focus of this thesis was on the development of a computer vision framework for improved RPAS operations, the various aircraft used were all commercially available and purchased by the Carleton UAV research group. The use of commercially available RPA for this work was intended to serve two purposes: to ensure reliable performance from the various aircraft test platforms and to allow the work performed in this thesis to be easily adapted by others for research and industrial applications. This section will outline information on each RPA and list the chapters in which respective work is described.

3.2.1 DJI Mavic 2 Pro

Figure 3.3: DJI Mavic 2 Pro RPA and DJI Smart Controller

The Mavic 2 Pro is a quadrotor RPAS platform that was marketed to the consumer-level operator for aerial imaging applications by SZ DJI Technology Co., Ltd. This aircraft was released in the 3rd quarter of 2018 and comes equipped with a Hasselblad imaging sensor that was specifically designed for small RPAS applications. This aircraft was acquired for the Carleton UAV research group for use as a standard imaging platform for data collection purposes in various research projects. Table 3.2 below summarizes the aircraft image sensor specifications as provided by the aircraft manufacturer.

Table 3.2: DJI Mavic 2 Pro image sensor specifications – (DJI, 2019)

Image Sensor	20 MP 1” CMOS Hasselblad
Number of Horizontal Pixels	5472 pixels
Number of Vertical Pixels	3648 pixels
Horizontal Field of View	77 degrees

This aircraft was used to acquire the aerial imagery data used in Chapter 4 of this thesis, which focuses on the development of a process to qualitatively assess the capabilities of an image sensor to meet image quality requirements of the RPAS mission.

3.2.2 Sky Viper Journey

Figure 3.4: Sky Viper Journey RPA and controller

The Sky Viper Journey is a small quadrotor RPA sold by Skyrocket that weighs under 250g. This aircraft is equipped with a manufacturer-customized version of the Arducopter software for flight control. This custom firmware limits the capabilities of Arducopter, which allows the aircraft to operate on a low-powered flight controller board. This RPA was used for the work presented in Chapter 6 to represent an off-the-shelf aircraft that is readily equipped with an image sensor, from which the video feed can be used to derive localization information of a known target within the image frame. As this image sensor was not documented by the aircraft manufacturer, the image sensor was tested by taking sample images, and the specifications of the image sensor are provided in Table 3.3.

Table 3.3: Sky Viper Journey image sensor specifications

Image Sensor Resolution	1 MP
Number of Horizontal Pixels	1280 pixels
Number of Vertical Pixels	720 pixels

3.2.3 3DR Iris+

Figure 3.5: 3DR Iris+ RPA

The Iris+ is a quadrotor RPA that was manufactured and sold by 3DRobotics. This aircraft was designed to be used as a modular platform, enabling researchers to perform modifications to the flight control system and airframe to accommodate small payload sensors. This aircraft served as a suitable platform to study the feasibility of integrating the proposed VPS through an independent system. The microcontroller used as the flight controller for this aircraft is the 3DR Pixhawk, which is commonly used for custom-designed RPA. This aircraft will serve as a representative small RPA for the work presented in Chapter 6 of this thesis.

3.3 Remotely Piloted Aircraft Flight Controls System

For RPA, the flight control system consists of a combination of hardware (processor unit, sensors, servos etc.) and software to enable both manual and autonomous control of the aircraft.

3.3.1 Flight Controller Hardware – Pixhawk

The Pixhawk is a commercially available, open-hardware microcontroller that was developed by 3DRobotics. It can be used to control a variety of autonomous vehicle platforms, ranging from aerial to ground-based robots. This versatile microcontroller comes readily equipped with a variety of sensors that enable it to perform tasks such as localization and waypoint navigation. It is also commonly equipped with a GNSS module to enable both the operator and aircraft to determine its location throughout the flight. The specifications of the Pixhawk processor and onboard sensors are as follows (Dronecode, 2019):

Processors

- 32-bit STM32F427 ARM Cortex M4 CPU with single-precision FPU, 180 MHz, 256 KB SRAM, 2 MB Flash
- 32-bit STM32F100 Cortex M3 CPU, 24 MHz, 8 KB SRAM

Internal Sensors

- ST Micro L3GD20H 16-bit gyroscope
- ST Micro LSM303D 14-bit accelerometer/magnetometer
- Invensense MPU 6000 3-axis accelerometer/gyroscope
- MEAS MS5611 barometer

Additionally, the Pixhawk can communicate with external computer systems via the MAVLink messaging protocol. This messaging protocol is used by external systems and sensors to send structured messages to the Pixhawk. These messages can contain a wide range of data, ranging from flight mode changes to desired waypoint and attitude commands.

3.4 Flight Controller Software – Arducopter

The flight controller software used by the Pixhawk is the Arducopter software, which is a branch of code from the larger Ardupilot project that is focused on the development of rotorcraft platforms. By utilizing the various sensors onboard the Pixhawk, it enables aircraft automation tasks such as attitude stabilization and waypoint navigation. Some of the key flight modes available are summarized in Table 3.4.

Table 3.4: Arducopter flight modes

Mode	Description
ALT_HOLD	The "ALT_HOLD" flight mode stabilizes the aircraft and attempts to maintain its altitude utilizing sensor data from the onboard barometer. When the control sticks of the RC controller is placed in the neutral positions, the aircraft will hold its altitude. This is useful for holding altitude in a GNSS-denied environment.
LOITER	The "LOITER" mode functions similar to "ALT_HOLD", except that the data collected by the GNSS module is used as well. With the additional information provided, this mode allows a rotorcraft to hold its position in 3-dimensional space.
GUIDED	This mode enables the flight controller to receive MAVLink commands from external sources; this allows for inputs of specific flight parameters, such as velocity, altitude, and heading. This mode utilizes the onboard sensors to achieve the desired flight parameters defined in the MAVLink command.
GUIDED_NOGPS	"GUIDED_NOGPS" is the equivalent of the "GUIDED" mode setting but is used for indoor flight applications. This mode removes the use of the GNSS module data when executing the input MAVLink commands. Input for this mode must be passed in the form of attitude commands.

3.5 Computer Vision Processing Hardware Platforms

Computer vision applications require comparatively higher processing power than most flight controllers such as the Pixhawk can provide. As such, additional hardware is required to perform the computational work presented in this thesis. This section will summarize the various computational platforms used for the experimental work presented.

3.5.1 Image Processing Workstation

The main image processing workstation to be used as the ground control station (GCS) for this thesis is the Lenovo Y520-15IKBN. This is a gaming laptop that houses a 7th generation Intel i7 processor and comes equipped with a dedicated Nvidia graphics card. It was used primarily for the image processing work presented in Chapter 4 and Chapter 6 of this thesis, and the dedicated graphics card enabled parallel processing used to speed up the neural network training work that is presented in Chapter 5. The specifications of the workstation are provided in Table 3.5.

Table 3.5: Lenovo Y520 system specifications

Processor	Intel Core i7-7700HQ 64-bit
Clock Speed	2.80 GHz, 3.80 GHz (Turbo Boost)
RAM	12 GB DDR 4
Number of Cores	4
Number of Threads	8
Dedicated Graphics Processing Unit	Nvidia GTX 1050 4GB
Clock Speed	1354 MHz
Number of CUDA Cores	640

3.5.2 Companion Computer

Certain RPA, such as the Iris+ platform presented in Section 3.2.3, do not have a readily available image sensor onboard. To mitigate this issue, a small companion computer system that can be equipped with a camera can be setup to enable video capture and image processing. For the applications presented in this thesis, the Raspberry Pi 3B+ was selected. It is a single-board computer that is commonly used for small robotics applications due to its favorable size and computing capabilities. It can be equipped with a

wide range of sensors; for the purposes of this thesis, it was equipped with an image sensor. The relevant specifications of the Raspberry Pi 3B+ are tabulated in Table 3.6.

Table 3.6: Raspberry Pi 3B+ system specifications - (Raspberry Pi Foundation, 2019)

Processor	Broadcom BCM 2837B0 64-bit
Clock Speed	1.40 GHz
RAM	1 GB LPDDR2
Number of Cores	4
Image Sensor	Raspberry Pi Camera v 1.3
Image Sensor Resolution	5 MP
H_{pix}	2592 pixels
V_{pix}	1944 pixels

3.6 Software Applications

In addition to the hardware, the data collected by the image sensor must be processed using specialized software packages. A combination of commercially available software libraries and programming interfaces were used throughout this thesis. This section will detail the various software used for the programming and image processing work presented.

3.6.1 Programming Language – Python

The programming language used for the work presented throughout this thesis is Python. Python is an open-source, redistributable programming language that can be used for both private and commercial applications. It is a highly versatile programming language, with access to various modules that can be used for analyzing and processing large volumes of data. The official version of Python can be obtained through Anaconda, an open-source distribution and package management software that comes with its own

interactive development environment (IDE), Spyder. For the programming work completed in Chapter 5 and Chapter 6, Python version 3.7 was used.

3.6.2 Image Processing Library – OpenCV

The OpenCV library is the image processing library selected to be the software development platform for the proposed computer vision framework. This is an open source computer vision and machine learning platform and is publicly accessible for commercial and research applications. OpenCV is natively developed in the C++ programming language, but has support interfaces for the Python, MATLAB, and Java programming languages; it is available for developing computer vision applications on the following operating systems: Windows, Linux, Android, and Mac OS. For the work presented in Chapter 6 of this thesis, the OpenCV library was used for the image processing experimental work conducted.

3.6.3 Neural Network Library – Keras

The Keras application programming interface (API) was used for the neural network work presented in Chapter 5. Keras is commonly used for the development of neural network models and runs natively on the Python programming language. It enables users to easily access popular machine learning platforms, such as TensorFlow.

3.7 System Benchmarking – Measuring Apparatus

As the proposed VPS is designed to provide position information of the aircraft in relation to a target, additional measuring apparatus were used for the work presented in Chapter 6 of this thesis. These tools are treated as the control for the experiments conducted, with the output of the VPS being compared to these for validation and benchmarking purposes.

3.7.1 LOMVUM LV-120M

The LOMVUM LV-120M Laser Distance Meter (LDM) is a "time of flight" based distance measuring device that is used for measuring linear distances. The device emits a laser beam at the target and measures the distance by calculating the amount of time taken for the reflected beam to reach the device. The relevant manufacturer specifications for its measurement range and accuracy is summarized in Table 3.7.

Table 3.7: LV-120M specifications

Specification	Value	Unit
Measuring Range	$0.2 \sim 120$	m
Measuring Accuracy	± 2	mm

3.7.2 RCbenchmark Otus Tracker

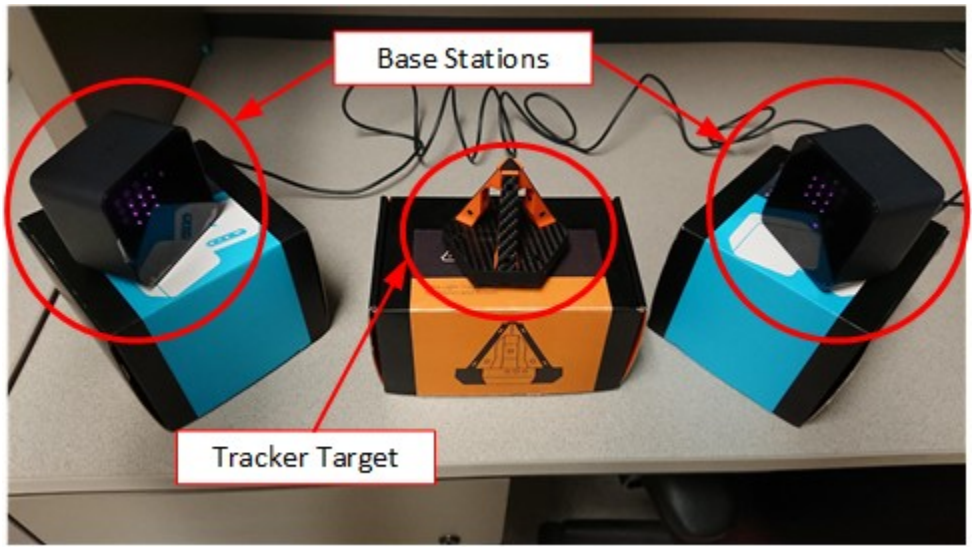

Figure 3.6: Otus Tracker system

The Otus Tracker hardware and Tracking Lab software is a 3D motion capture system that was developed by RCbenchmark (previously Tyto Robotics Inc.). It consists of a lightweight tracker and two base stations and is built upon the "room-scale" technology that was designed for virtual reality applications. By utilizing two base stations that actively

emit infrared pulses, the system can calculate and determine the tracker's position. The manufacturer specifications for the system are provided in Table 3.8.

Table 3.8: Otus Tracker system specifications – (RCbenchmark, 2019)

Specification	Value	Unit
Position Precision	± 1	mm
Position Accuracy	± 1	cm
Orientation Precision	± 1	degree
Wireless Communication Rate	250	Hz
Pose Update Rate	$0.1 - 400$	Hz
Tracking Area	5×5	m^2

For the purposes of the experiments conducted with system, the pose update rate was set to a constant 20 Hz.

Chapter 4: Image Sensor Qualitative Study

Many consumer-level RPAS found on the market today come equipped with imaging sensors. The advancement in electronics has enabled sensor manufacturers to reduce the size of the physical hardware while also increasing the resolution of the imagery data collected. In recent times, a common use for RPAS imagery data is to perform photogrammetry for mapping applications. By correlating the known geo-location of the aircraft with the multiple images collected in-flight, it is possible to create ortho-mosaics of the surveyed area to produce maps. Another use of RPAS imagery data is to create 3D point clouds of a scanned building or structure. This process takes into account the position of the aircraft in relation to the scanned object and utilizes the process of Structure-from-Motion (SfM) (Ullman, 1976) to identify matching points between the images collected to create a 3D point cloud of the structure. Image sensors with larger resolutions provide the user with the ability to capture more detailed imagery data. However, this also leads to larger data file sizes, which can limit the number of images and/or videos collected per flight. Additionally, greater volumes of data storage (either offline or cloud-based) would be required to store the previously collected data for post-processing purposes.

This chapter presents the steps required to determine the suitability of an image sensor to fit the specific imaging needs of an RPAS to complete its intended imagery data collection application. This qualitative analysis method can then be used to assist RPAS manufacturers to select suitable image sensors to fit their intended application, as well as to provide RPAS operators with the ability to assess the capabilities of an RPAS imaging sensor to meet their mission needs. This will ensure that the RPAS operator can maximize

the use of their onboard and offboard storage while maintaining key visual information necessary to achieve the desired level of image quality for their application.

4.1 Image Sensor Specifications – Manufacturer-provided Metrics

Image sensors are complex electronics that can vary significantly due to the large number of components that they house. Components such as the number of lenses, shutter type, sensor format, and more all affect the performance of an image sensor in a wide range of lighting conditions. Due to the large number of metrics that affect the imaging quality of a sensor, it is important to identify some high-level metrics by which image sensors can be quantitatively compared. These quantitative metrics include the output image resolution, image sensor size, and the effective field of view of the image sensor; these are discussed in further depth in the subsections below.

4.1.1 Image Sensor Pixel Resolution

Image sensor pixel resolution refers to the number of pixels that are present in the imaging frame, and by extension the "level of detail" that it contains. This specification is provided by the image sensor manufacturer and can be found in two popular conventions - the total number of pixels and the number of pixels in a horizontal and vertical line within the image frame. The total number of pixels can be determined by multiplying the number of horizontal and vertical pixels.

Table 4.1: Common definitions of image sensor resolution

Total number of pixels:	Commonly denoted in units of megapixels (MP)
Number of pixels in a line:	Commonly denoted in the form of $H_{pix} \times V_{pix}$

Image resolution is an important sensor metric, as it provides the user with information on the level of detail that can be captured by the imaging technology. The

larger the resolution of an image sensor, the greater the level of detail stored in the visual data collected.

4.1.2 Sensor Size

Sensor size is another important metric when considering the performance of an image sensor. The sizing information provided by manufacturers is commonly expressed in the form of the diagonal length in inches. This metric is important to consider when looking to use the imaging sensor in very specific lighting conditions. A larger sensor size usually leads to better performance in low light conditions.

4.1.3 Field of View

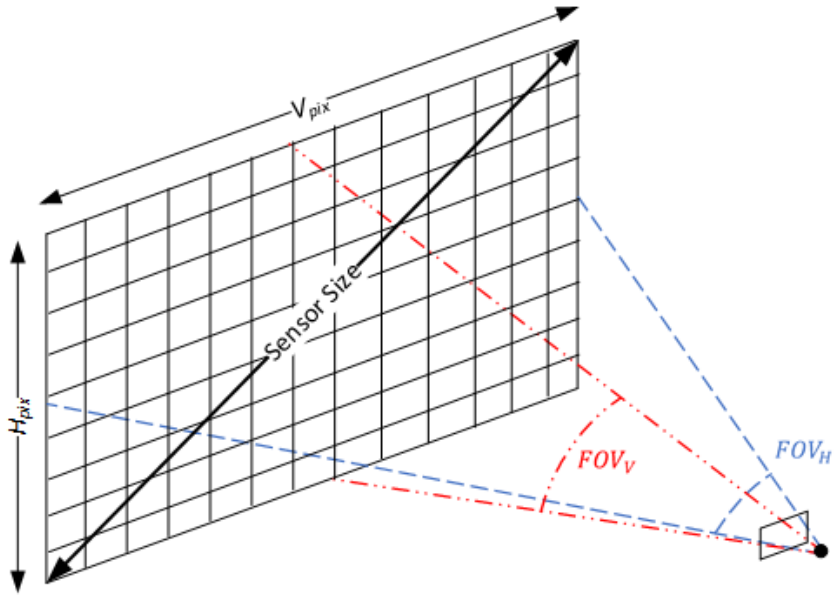

Figure 4.1: Image sensor specifications illustration

The Field of View (FOV) metric of an image sensor provides information on the effective viewing angle of the image frame from the sensor. Figure 4.1 illustrates the definition of the FOV of an image sensor in the horizontal plane and vertical plane. Most manufacturers provide information on the horizontal FOV of an image sensor. With the

resolution of the image sensor known, the aspect ratio can be determined and used to derive the vertical FOV of the image sensor utilizing Equation 4.1 and Equation 4.2.

$$Aspect\ Ratio = \frac{H_{pix}}{V_{pix}} \quad \text{(Eq 4.1)}$$

$$FOV_V = 2 \cdot \tan^{-1}\left(\frac{\tan\left(\frac{FOV_H}{2}\right)}{Aspect\ Ratio}\right) \quad \text{(Eq 4.2)}$$

When comparing image sensor FOV metrics, it is important to note that a larger FOV does not result in more visual information from an image processing perspective. A target viewed using an image sensor with a smaller FOV will occupy a larger portion of the resulting image. An example of this would be the comparison between a zoom lens and a wide-angle lens – the image taken with a wide-angle lens, which has a large FOV, will capture a wide perspective of the objects in front of the camera; conversely, a zoom lens, which has a smaller FOV, will reproduce a narrower perspective of the objects, allowing for greater focus on specific subjects in the image frame.

4.2 Ground Sampling Distance

The manufacturer-provided metrics discussed in Section 4.1 alone do not quantify the field performance of the sensor; most applications of RPAS equipped with image sensors revolve around enabling the operator to either monitor or identify specific targets while in use. To assist the operator in determining the suitability of an image sensor in capturing the desired imagery data within the image frame, Gundlach recommends the use of Ground Sampling Distance (GSD) (Gundlach, 2012). The GSD of an image sensor can be determined using the image sensor specification and the position of the image sensor in relation to the desired target.

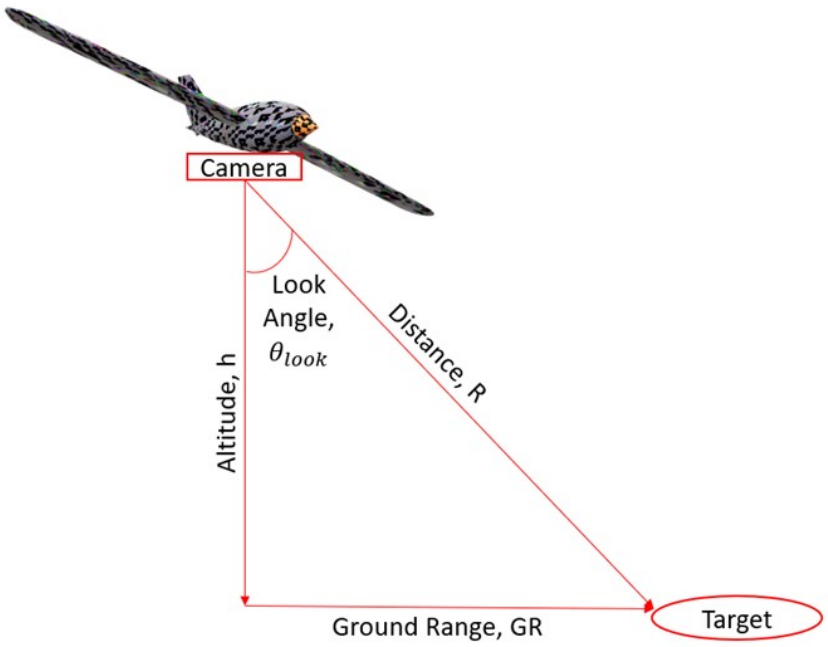

Figure 4.2: *GSD* calculation geometry - aircraft position

Figure 4.2 displays the geometry of the values related to the position of the aircraft that is required in determining the *GSD* of an image sensor. When operating an RPAS, the altitude of the aircraft, h, is known, and the look angle of the image sensor, θ_{look}, can be set by the user. Equation 4.3 and Equation 4.4 can then be used to determine the horizontal and vertical GSD, respectively.

$$GSD_H = 2 \cdot \tan\left(\frac{FOV_H}{2 \cdot H_{pix}}\right) \cdot R \quad \text{(Eq 4.3)}$$

$$GSD_V = \frac{2 \cdot \tan\left(\frac{FOV_V}{2 \cdot V_{pix}}\right)}{\cos(\theta_{look})} \cdot R \quad \text{(Eq 4.4)}$$

The horizontal and vertical GSD values provide information on the real-world distances that each pixel in the captured image frame represents. By multiplying GSD_H and

GSD_V by H_{pix} and V_{pix} respectively, the total distances covered by the horizontal and vertical image plane can be calculated.

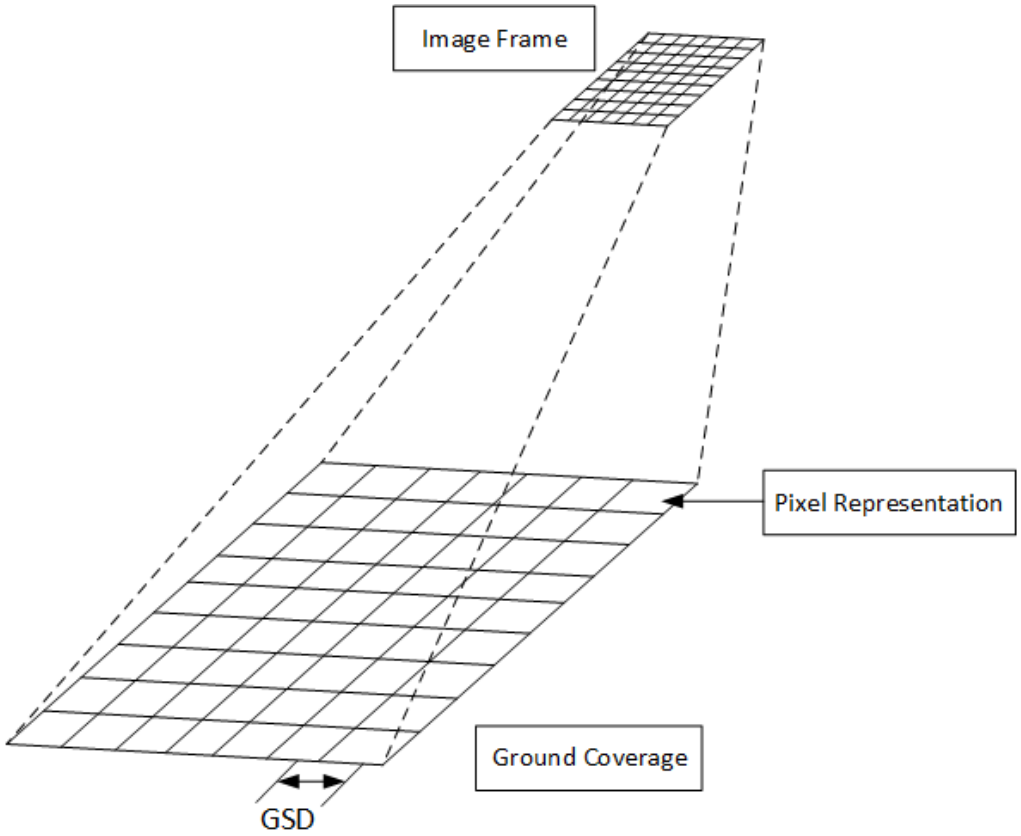

Figure 4.3: Image plane and ground coverage correlation

With the calculated GSD values, there is a correlation between each pixel and the real-world measurements. The calculated GSD_H and GSD_V values for both the horizontal and vertical axes, respectively, can then be used to measure the estimated size of objects within the image frame.

4.3 Image Sensor Quality Metric – Johnson criteria

The Johnson criteria is a commonly used image quality metric that was introduced in 1958 by Johnson in his qualitative study on image intensifier technologies (Johnson, 1958). This image quality metric allows the user to predict the performance of an imaging

system in enabling a trained human observer to perform one of three types of image classification tasks: detection, recognition, and identification. The Johnson criteria uses the known resolution of an image sensor, together with the characteristic dimension of the target, to determine the probability of classifying said target within the image frame. A pair of line pixels, referred to as a "cycle", is needed in order to provide the necessary visual information for a human observer to be able to make an inference on the target object.

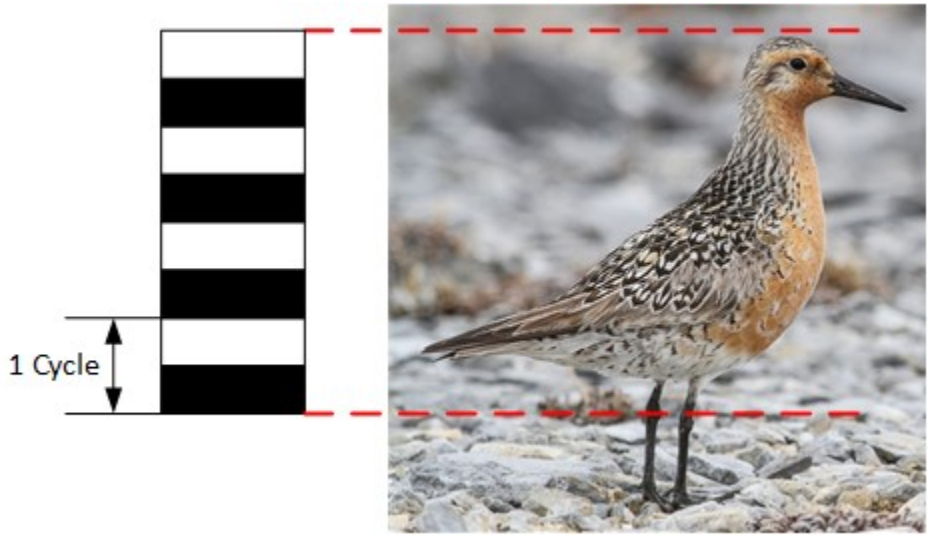

Figure 4.4: Johnson criteria "cycle" definition – adapted from (Davies, 2013)

Each of the three classification categories mentioned previously are associated with increasing image resolution requirements to provide greater visual detail. A larger number of cycles results in greater visual information being provided to the visual observer performing the classification task. Table 4.2 provides a definition for each of the classification tasks and the associated number of cycles on target required for a 50% probability of correctly performing said task.

Table 4.2: Johnson criteria image classification task summary

Classification Task	Number of cycles, N_{50}	Task Definition
Detection	0.75	Determine the presence of an object
Recognition	3.0	Differentiate between type of object (inter-class)
Identification	6.0	Determine the specific class of the object (intra-class)

To calculate the probability of completing a specific visual task, the Johnson criteria requires information on both the target and the image sensor's GSD. A characteristic size for the target, d_c, must be determined, as well as the average GSD of the image sensor. These values can then be used to determine the number of cycles, N, that the target would present itself as in the image frame. Subsequently, the probability of a human observer successfully performing the desired visual task can then be computed by using Equation 4.8, substituting the previously calculated N value and the N_{50} value for the desired level of image classification.

$$d_c = \sqrt{H_{tgt} \times W_{tgt}} \quad \text{(Eq 4.5)}$$

$$GSD_{avg} = \sqrt{GSD_H + GSD_V} \quad \text{(Eq 4.6)}$$

$$N = \frac{d_c}{2 \times GSD_{avg}} \quad \text{(Eq 4.7)}$$

$$P(N) = \frac{\left(\frac{N}{N_{50}}\right)^{2.7+0.7\left(\frac{N}{N_{50}}\right)}}{1 + \left(\frac{N}{N_{50}}\right)^{2.7+0.7\left(\frac{N}{N_{50}}\right)}} \quad \text{(Eq 4.8)}$$

4.4 Experimental Testing and Results

(a) (b)

Figure 4.5: Least Sandpiper - (a) real specimen (Lipton, 2016) (b) 3D model

To quantify the results of the Johnson criteria, a comparison between the three levels of image classification is presented. This was done in the form of a case study for an image sensor qualitative study that was performed for the DJI Mavic 2 Pro. In this case study, the aircraft is being used to assist in the monitoring of endangered Arctic shorebirds. The goal of this case study is to determine the capabilities of the image sensor to provide enough visual information to enable a visual observer to perform the task of identification. The Least Sandpiper (*Calidris minutilla*) is one of the targets to be monitored with a RPAS – it represents the smallest bird of the nine species of interest. The estimated size of the Least Sandpiper is summarized in Table 4.3. To facilitate the testing process, a 3D printed model of the target was produced and painted to closely imitate the live specimen; a close-up of the target is provided in Figure 4.5.

Table 4.3: Least Sandpiper body dimensions

Body Length, H_{tgt}	$78\ mm$
Body Width, W_{tgt}	$31\ mm$
Characteristic Size, d_c	$50\ mm$

Using the image sensor information summarized in Section 3.2.1, the process of determining the GSD values presented in Section 4.2 is used to determine the operational limitations of the aircraft in terms of altitude and distance away from the target. A contour map with the probability of performing the specified visual task can be produced with the altitude, h and the ground range, GR being the two varying flight parameters. With this information, a suitable flight altitude can be calculated at which a 100% probability of successful detection by a human observer can be determined. For simplicity, the aircraft is operated directly above the target. This same step is repeated for both the task of recognition and identification. The relevant information pertaining to the desired position of the aircraft in relation to the target, the respective GSD_{avg} values, and the probability of completing the visual task with a 50% success rate are summarized in Table 4.4.

Table 4.4: Aircraft flight altitude and GSD_{avg} values

Classification Task	**Altitude, h**	**GSD_{avg} Value**	**Probability**
Detection	$10\ m$	$2.56\ mm/pix$	100.00
Recognition	$6\ m$	$1.53\ mm/pix$	99.99
Identification	$4\ m$	$1.02\ mm/pix$	99.96

4.4.1 Test 1 – Detection

The task of detection represents the highest level of classification; it is to determine if the target is present within the image frame. The contour plot for the probability of detection is presented in Figure 4.6. This same contour map can also be used to estimate the probability of performing the task of detection at varying aircraft positions in relation to the target.

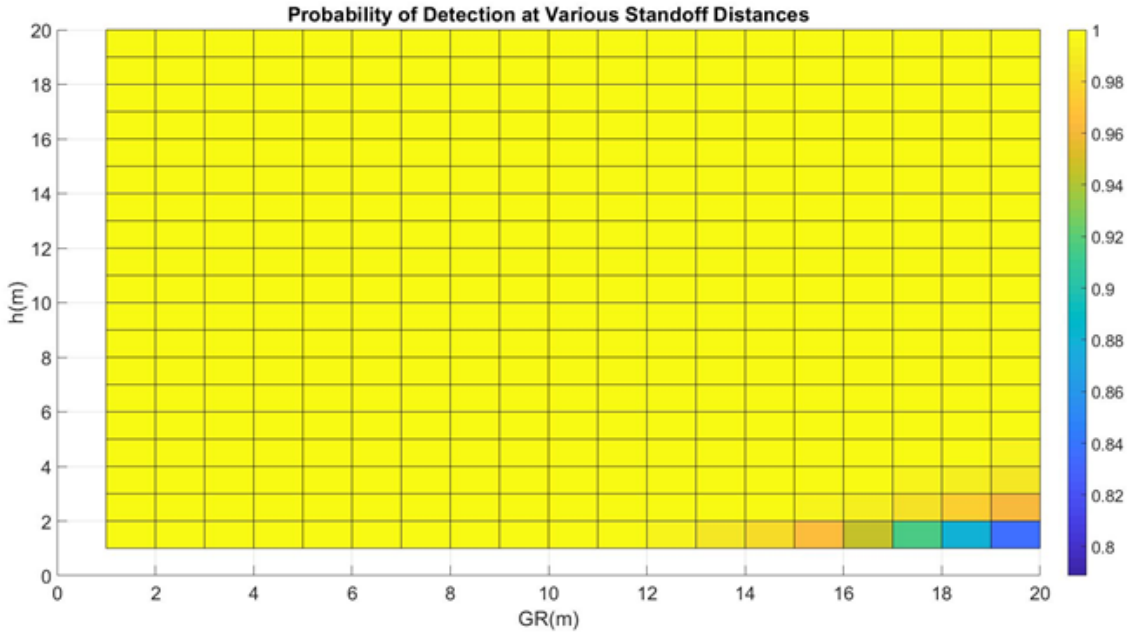

Figure 4.6: Contour plot for probability of detection of the target

Figure 4.7 shows the original image on the left that was collected with the RPAS at an altitude of $h = 10\,m$. The right image is the zoomed-in image of the Least Sandpiper model that was used for the testing described in this chapter. The visual information that can be extracted from the enlarged image is sufficient to ascertain that there is an object in the image frame. The heavy pixilation of the image is due to the large GSD_{avg} value associated with the flight altitude. A larger GSD_{avg} value results in fewer pixels on target – this leads to less visual information of the target being stored.

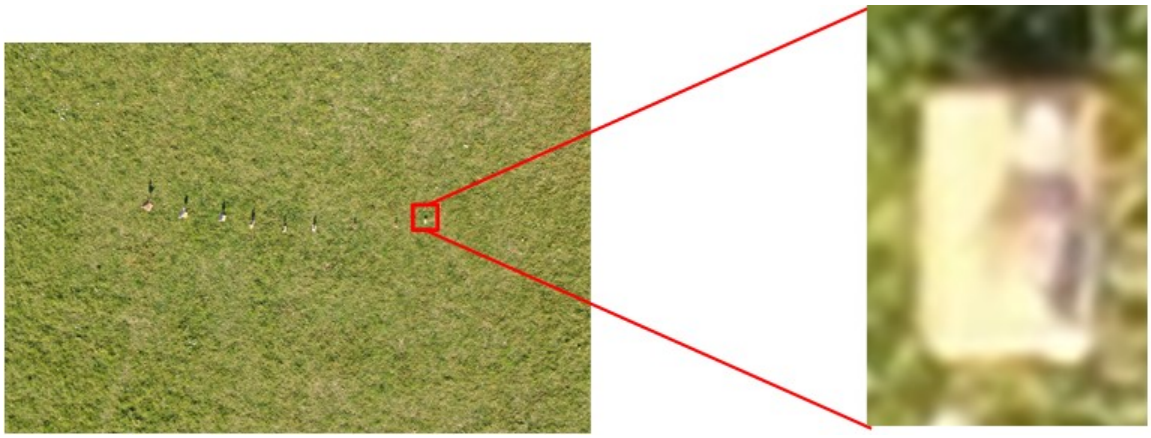

Figure 4.7: Johnson criteria test for probability of detection (h = 10 m)

4.4.2 Test 2 – Recognition

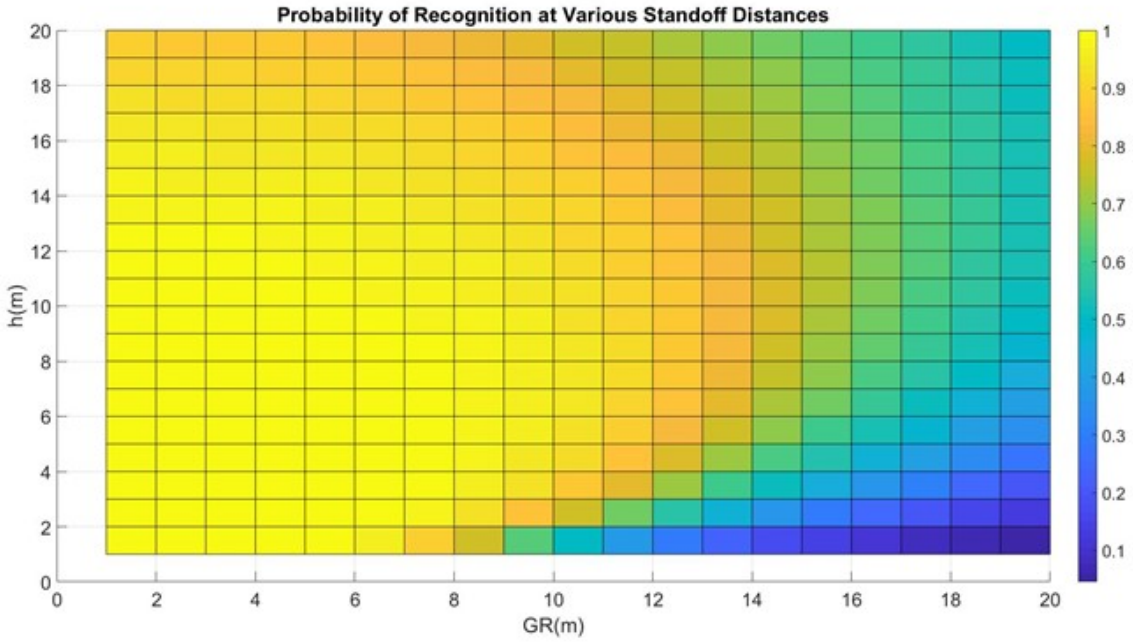

Figure 4.8: Contour plot for probability of recognition of the target

The task of recognition involves further differentiation between the class of the target. Figure 4.9 provides both the initial image captured at $h = 6\,m$ and the enlarged image of the target. The smaller GSD_{avg} value at this altitude results in more pixels on target, which provides the visual observer with more detailed imagery data to perform the classification task. At this altitude, it is possible to see clear coloration differences between

the various body parts of the bird; the brown dorsal coloration can more easily be distinguished from the lighter colored head and tail. Also, the feet of the bird are more clearly visible, which provides further information on the pose of the target.

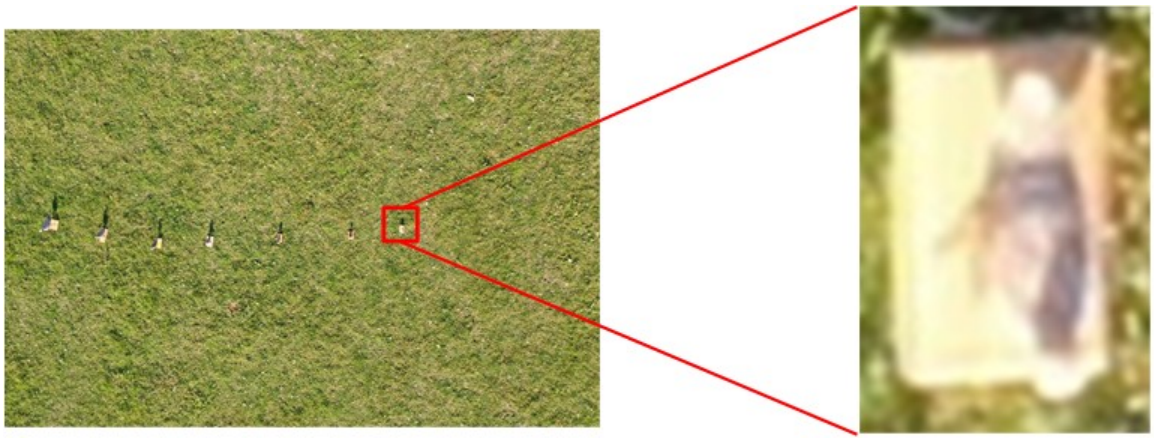

Figure 4.9: Johnson criteria test for probability of recognition (h = 6 m)

4.4.3 Test 3 – Identification

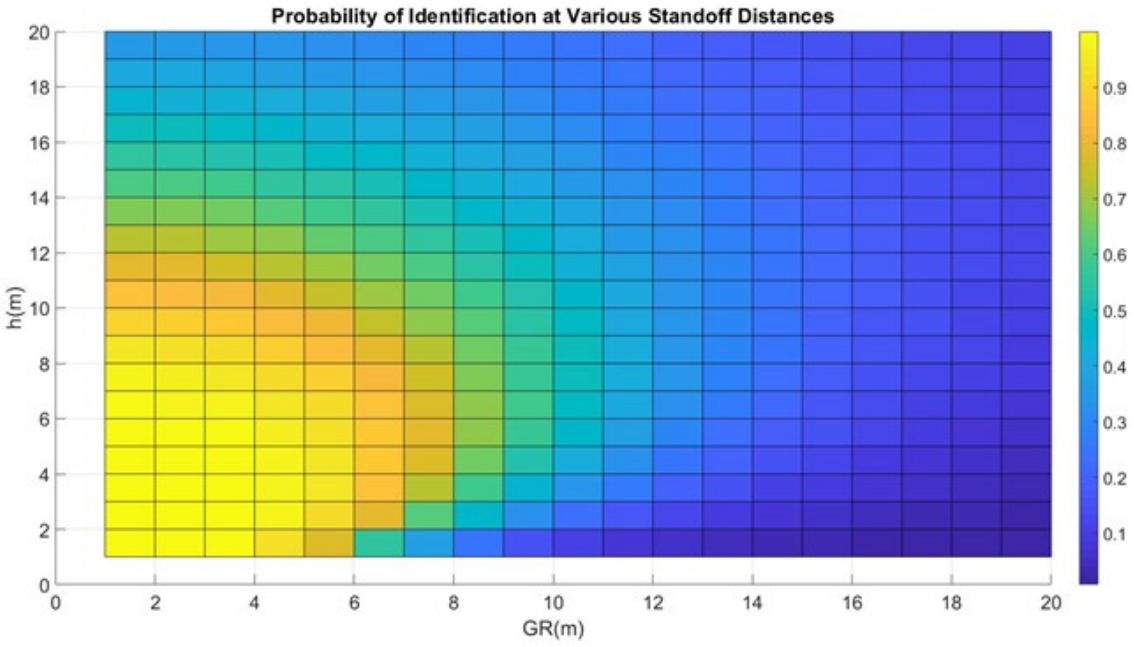

Figure 4.10: Contour plot for probability of identification of the target

For image identification applications, greater visual information is necessary. Intra-species identification is the most difficult task due to the need for visual data on key

identifiers that must be captured by the imaging sensor. Key features needed to enable the task of identification include repetitive patterns, further distinction between coloration, and the presence of specific features that are unique to the target.

Figure 4.11: Johnson criteria test for probability of identification (h = 4 m)

The enlarged image in Figure 4.11 of the target shows clearer coloration, enabling the visual observer to more clearly discern the patterns on the dorsal. The spotted pattern is more clearly visible than in the previous test, and the larger spots can also be individually counted. The outline of the target's body is also more clearly distinguished from the wood plank that the specimen is mounted to. The general shape of the white coloration on the tail could also be used for identification purposes. Additionally, the wording that is located on the lower side of the wood plank is also noticeable; however, it is unintelligible at this altitude.

4.4.4 Estimation of Target Size Using GSD_H and GSD_V

For this experiment, the DJI Mavic 2 was flown at a known altitude ($h = 5\ m$) and camera pose ($\theta_{look} = 0$ deg) above the targets for a still image shot. The horizontal and vertical GSD values are calculated and presented in Table 4.5. Bounding boxes around the targets are individually declared, and the corresponding pixel coordinates associated with

the bounding box are used together with the GSD values to obtain estimations of the measured lengths in real-world units. Figure 4.12 is the original image file used for this test.

Table 4.5: *GSD* values for DJI Mavic 2 Pro at $h = 5\ m$ and $\theta_{look} = 0\ deg$

Horizontal GSD, GSD_H	1.23 mm/pix
Vertical GSD, GSD_V	1.35 mm/pix

Figure 4.12: Target lineup (L to R): Hudsonian Godwit, Greater Yellowlegs, Black-bellied Plover, Red Knot, Lesser Yellowlegs, Dunlin

As the dimensions of the 3D printed bird models are known, a percent difference can be determined for each individual measurement performed. For this test, only the body length of the targets is compared with the digitally measured lengths. The results of this comparison are tabulated in Table 4.6.

Table 4.6: Comparison of measured and actual body lengths for specimens

Specimen	Measured Body Length (mm)	Actual Body Length (mm)	Percent Difference (%)
Hudsonian Godwit	205	235	12.6
Greater Yellowlegs	204	207	1.8
Black-bellied Plover	217	230	5.6
Red Knot	176	175	0.3
Lesser Yellowlegs	156	154	1.4
Dunlin	90	114	21.6

As seen from the results presented, the measured values differ from the actual measurements. This is likely due to the manual process of declaring a bounding box around the target, which can be prone to error. In combination with the varying resolution of the viewing screen, it is difficult to declare the bounding box at the exact locations of the respective horizontal and vertical line pixels where the target exists. To minimize errors in estimation, the development of a computer vision algorithm to automatically identify and define the bounding box around the individual targets can be used.

4.5 Conclusions

This chapter presented a method to evaluate an image sensor for applications towards more detailed visual data collection. By utilizing the Johnson criteria and known image sensor specifications, it is possible to forecast the performance of the sensor in providing varying levels of detail in the collected imagery data. Based on the desired image classification task to be conducted, it is possible to determine the most suitable image sensor given preset flight parameters, or vice versa. Additionally, the use of the GSD concept further allows for estimation of target sizing.

Chapter 5: Image Processing using a Convolutional Neural Network

The development of machine learning algorithms has enabled the use of CNNs for image processing applications. Being a class of Artificial Neural Networks (ANN), CNNs can be used to perform several types of image processing, which are depicted in Figure 5.1. Each of these tasks are further summarized in Table 5.1.

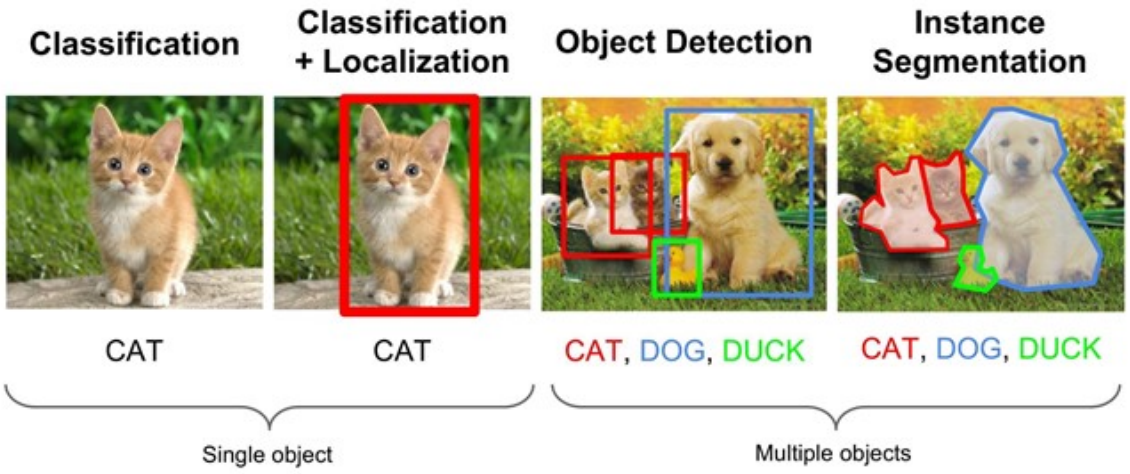

Figure 5.1: CNN image processing tasks visualization (Ouaknine, 2018)

Table 5.1: CNN image processing tasks

Image Processing Task	Definition
Classification	Prediction of a single object class with the highest probability of existing within the image frame.
Classification + Localization	Like classification, but a bounding box for the location of the target is also predicted.
Object Detection	Prediction can be made on multiple object classes that exist in the image frame and their respective locations.
Instance Segmentation	Like object detection, but the shape outlines of the objects are predicted rather than a generalized region.

The general framework of a CNN model is depicted in Figure 5.2; it is subdivided into two major segments – feature extraction and classification. These two components

work together to perform the task of analyzing an input image and provide the necessary prediction output based on the CNN's application. The feature extraction portion is comprised of a series of convolutional layers and pooling layers – these enable the CNN model to identify and extract features within the input image while simultaneously reducing the information passed on to the next layer. The classification layers consist of multiple fully connected layers that perform the task of correlating the information found in the feature extraction layers to the output classes. The final output of a CNN is the model's prediction overlaid on the input image.

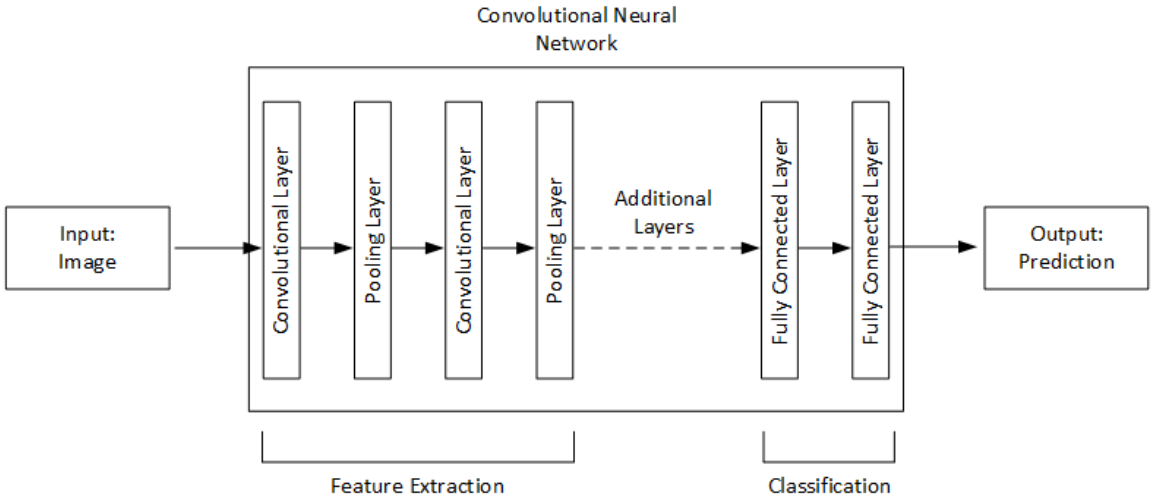

Figure 5.2: CNN framework

The size and complexity of a CNN is determined by the developer, as the combination of additional layers and its associated user-defined parameters can significantly affect the performance of the model. It is also important to note that a CNN requires a suitably large dataset to serve as an input for training and validation purposes. The training process is time consuming for large datasets, as it sorts through the images to extract key features to be used for making the necessary predictions on new images. As

such, many variables must be considered when looking to develop a CNN for an application-specific case.

Although robust CNNs for general applications, such as facial recognition and high-level image classifications are readily available for open-source usage (as discussed in Chapter 2), these models do not perform well with new, unclassified objects. As such, this chapter will focus on the development of a CNN for mission-specific image processing applications. Due to the highly customizable layout of a CNN, developing a deeper understanding of how each user-defined parameter affects the performance of the model is important. This knowledge can then be used to determine the most suitable parameters to modify when developing a CNN model, to ensure that its performance meets the required prediction accuracy.

5.1 CNN Operating Theory

To understand the application of a CNN for image analysis, it is first important to understand how a computer processes and derives information from an image frame. When processing an image frame, a computer is unable to perform image analysis the same way a human does; it is unable to directly perceive the visual information provided by the image sensor to the vision system. Where a human can instantly identify lines and edges to determine the shape of an object, a computer must perform computational tasks that require a finite computing time to make these same distinctions. When an image is collected and stored in a digital medium, it is commonly recorded either in RGB (3-color channel) or in grayscale (1-color channel). Each pixel in the image frame will contain a pixel intensity value ranging between 0 and 255.

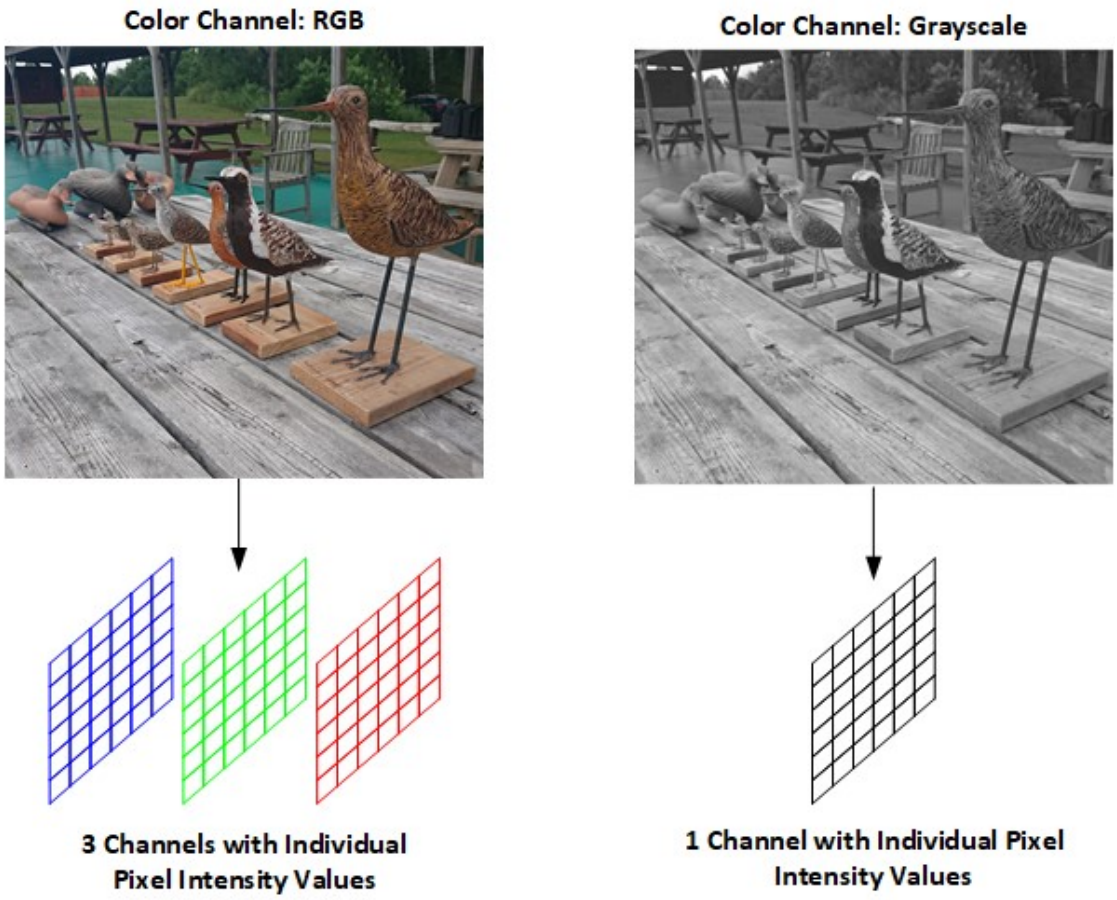

Figure 5.3: Comparison of RGB and grayscale images

For example, an image stored in RGB has three individual channels, each with their respective pixel intensity values for that color stored within the digital image file, which can then be used to recreate the image when viewed on a screen. Figure 5.3 shows a simplified representation of a typical channel grid; each channel will have a data grid of dimensions H_{pix} and V_{pix} along the horizontal and vertical axes, respectively – each grid square represents a single pixel. The size of the data grid is determined by the image sensor resolution, which was discussed in Chapter 4 of this thesis. For the examples used to illustrate the operations of a CNN, a simplified setup is used where the pixel values will be limited to a value of either 1 or 0, to represent the presence of a pixel within a simplified 1-dimensional plane.

5.1.1 Feature Extraction – Convolutional Layer

The first step when processing an image through a CNN is to apply pre-trained filters to identify key characteristic features of the desired object to be classified within the image frame. These filters are automatically determined during the training portion of developing a CNN, and the accuracy of the model can be affected by the size of the filters used during training. A commonly used filter size is of dimensions $3 \times 3\ pixels$. Figure 5.4 is a simplified illustration of how a CNN filter is derived from a set of three training images.

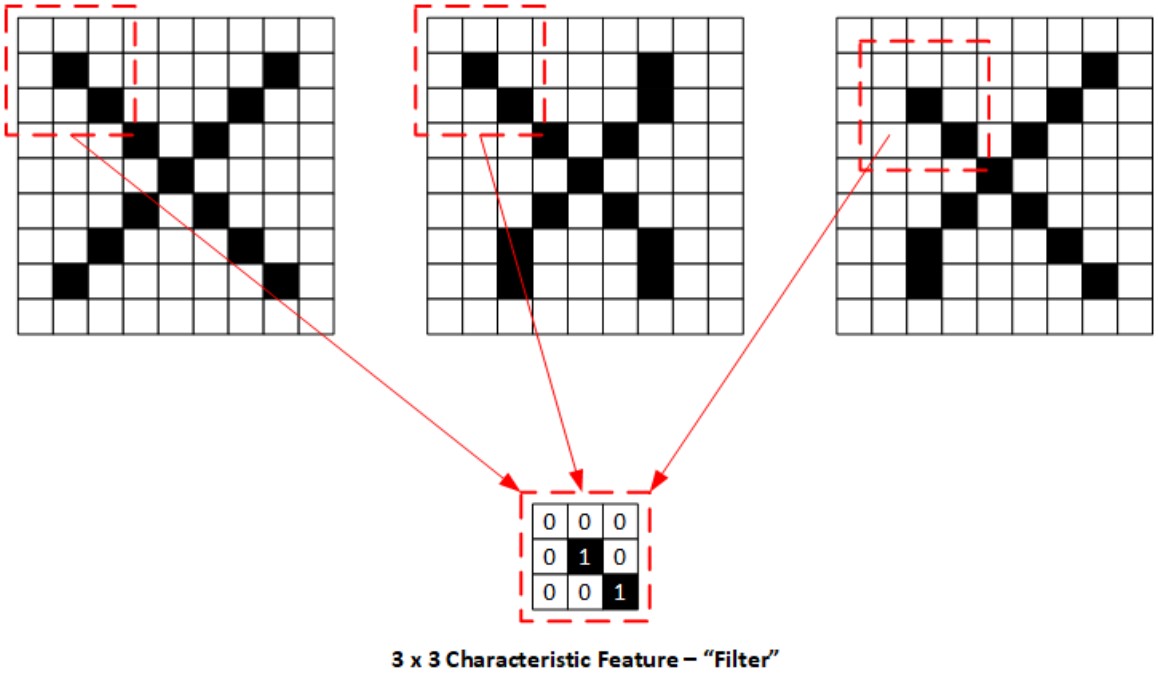

Figure 5.4: Identification of characteristic feature from training images

When a CNN is undergoing training, it identifies characteristic features from the input images. For the set of "X" images above, the $3 \times 3\ pixels$ filter is passed through each image separately, and the CNN will identify characteristic features that are present in all of the input images, such as the example shown in Figure 5.4. This filter size is defined when developing the convolution layer of the CNN; more complex CNNs vary the size of

filters used between each convolution layer to capture varying levels of detail at each convolution operation. When the trained model is applied to a new input image, this filter is applied to it; if this characteristic feature is present, it will be classified as a positive identification for "X". However, having just one filter is insufficient, as this can lead to many false-positive identifications. As such, multiple filters are trained and applied to a CNN to reduce erroneous classification. Figure 5.5 displays the mathematical process that occurs when the filter is applied to the new image from the previous example.

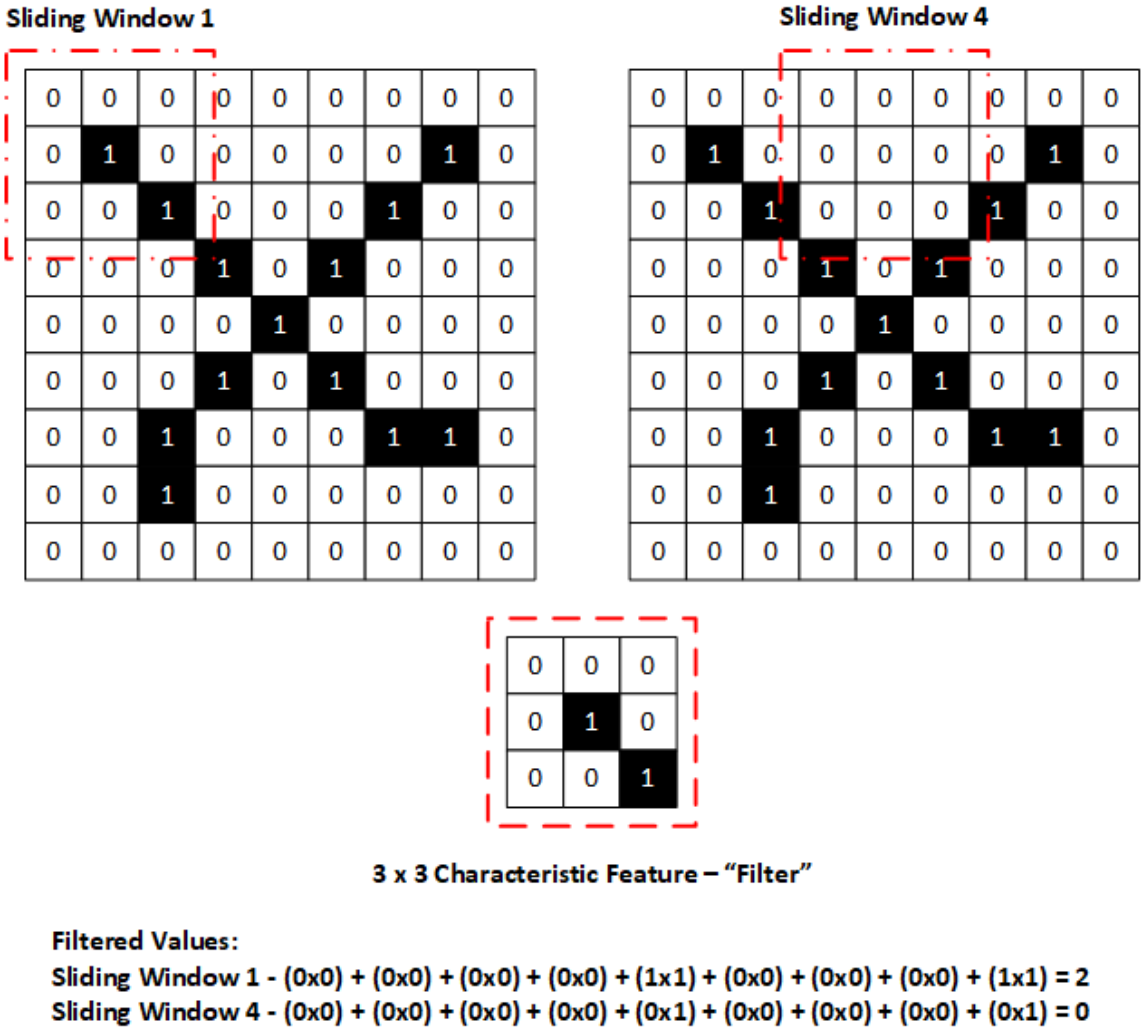

Figure 5.5: Sliding Window process for convolution operation

This operation is an element-wise multiplication between the trained filters and a "sliding window" of the same dimensions – this process is known as convolution. The convolution step provides information on the location of the characteristic features within the image frame while simultaneously reducing the amount of information to be processed by the ensuing layer. The process of moving the "sliding window" is governed by a parameter known as stride length; a commonly used value is 1, signifying a movement of a single row or column of pixels, to minimize excess loss of information between layers. The result of the convolution operation is displayed in Figure 5.6, which highlights the reduction in dimension of the data grid.

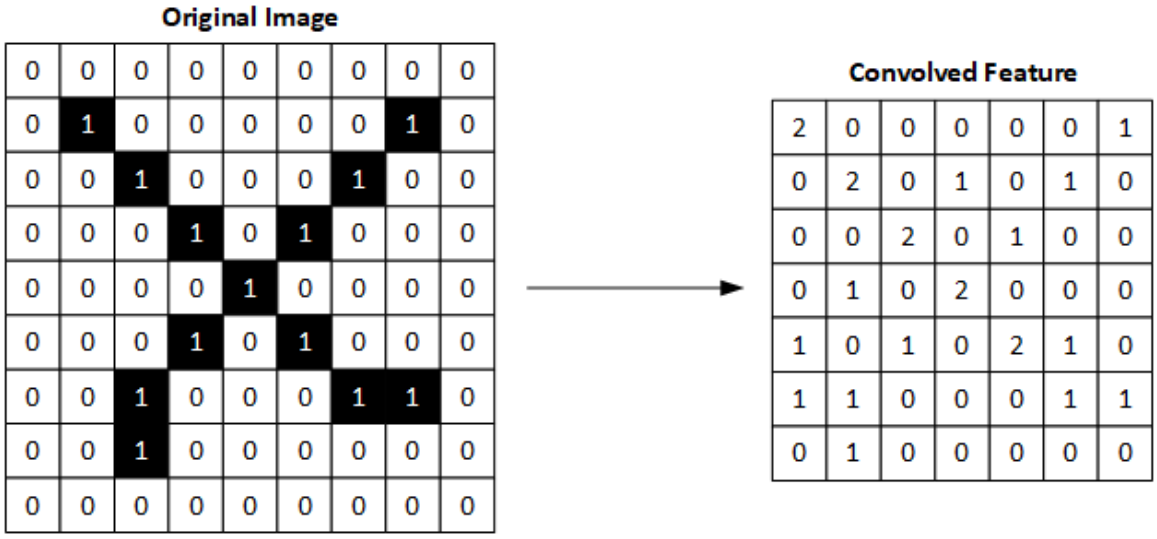

Figure 5.6: Convolved feature output

The first convolutional layer will capture the low-level details of the image, such as characteristic edges and color gradients associated with the object to be classified. However, as discussed at the beginning of this chapter, a CNN is comprised of multiple different layers. Each subsequent convolutional layer applied will capture increasingly higher-level features within the image due to the data reduction properties of the operation.

5.1.2 Feature Extraction - Pooling Layer

The next layer in a CNN is known as the pooling layer – it reduces the image data between convolutional layers. The pooling layer functions like the convolutional layer in that a "sliding window" of a specific size is passed through the entire image. One of two types of operations may be used to further reduce the image data – these are summarized in Table 5.2 and is visually represented in Figure 5.7:

Table 5.2: Pooling layer type comparison

Pooling Layer Type	Operation
Max Pooling	A $n \times n$ sliding window is passed through the entire image matrix, and the window region is reduced to the largest pixel value found.
Average Pooling	A $n \times n$ sliding window is passed through the entire image matrix, and the window region is reduced to the average pixel value.

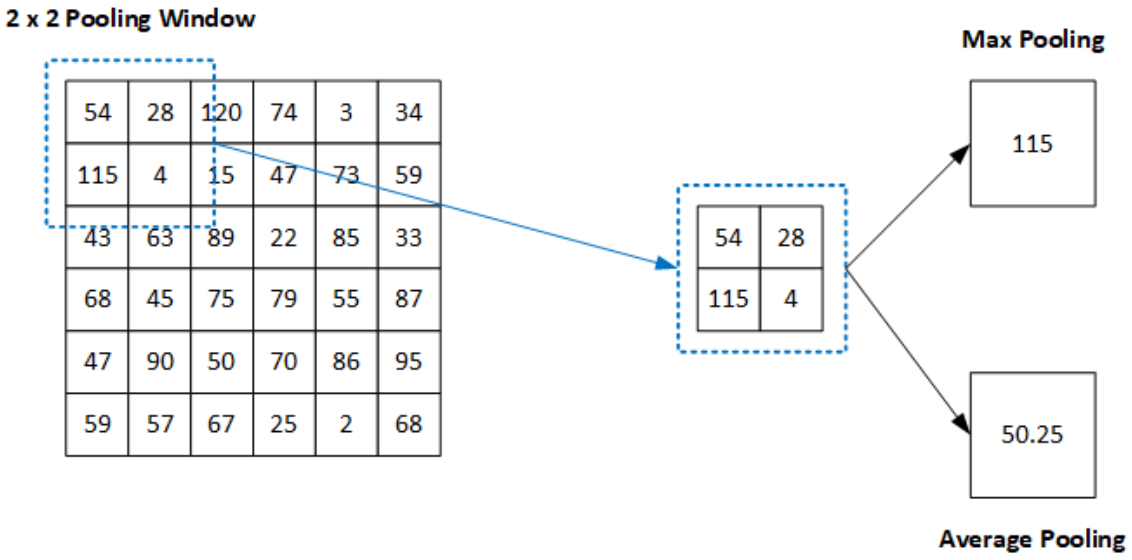

Figure 5.7: Comparison of max and average pooling layer operations

As the operation only involves either taking the maximum or the average pixel value of each individual region, this layer is helpful in extracting highly dominant features that are invariant to positional or rotational changes. A max pooling layer captures the most dominant feature present in the search region, while an average pooling layer captures the average of the features in the search region. As such, from a data handling perspective, a max pooling layer is more beneficial for preservation of information, as the dominant features in the search region are passed on to the following layers while still assisting in the data reduction throughout the feature learning process of a CNN.

5.1.3 Classification - Fully Connected Layer

The fully connected layer in a CNN functions like those of a standard neural network. These layers perform the task of correlating the characteristic features that were identified by the feature extraction layers in the CNN. Each of the identified filters are assigned a certain weightage during the training portion of development; these weights are readjusted between training steps as the CNN is provided with more randomized images from the training set. These weights are then used by the trained model to predict the object class as an output for a given input image.

5.2 CNN Model Training

As discussed in the beginning of this chapter, a CNN is highly customizable, and is usually dependent on the desired image classification task and number of predictive outputs. For example, a binary classification problem may be possible with the use of a less complex CNN framework to perform the necessary predictive output of 0 or 1, representative of the two objects being classified. However, a multi-class classification model may require a more complex framework, consisting of more layers to identify the

necessary characteristic features to be able to accurately make a prediction for the model output, as seen with the pre-trained models discussed in Chapter 2.

5.2.1 CNN Model Layer Setup Parameters

A CNN is a predictive model that is built based upon the dataset that it is provided with during the training and validation steps. As such, there are a variety of user-defined parameters that must be considered when constructing a CNN. These parameters affect how the various layers of a CNN processes the information it is provided and can influence its overall performance in making accurate predictions. A brief summary of the function of each of these parameters is provided in Table 5.3.

Table 5.3: Summary of user-defined CNN model training parameters

CNN Training Parameter	Description
Filter Size (Convolutional Layer)	This parameter allows the user to define the size of the $n \times n$ filter window used to identify characteristic features and perform the convolution operation.
Number of Filters (Convolutional Layer)	The total number of filters to be developed by the CNN during the training step; these are then used when processing new input images to make a prediction.
Input Size (Input Layer)	The resolution at which the training images are passed through the CNN; for reduction in training time purposes, compression is usually applied to the input images.
Pooling Size (Pooling Layer)	Size of the $n \times n$ window used in the pooling layer to reduce the amount of information passed between layers.
Batch Size	Total number of random samples from the dataset to be used for one training cycle; generally used for system memory management purposes, with larger sizes leading to longer training times.
Epoch	Term used to define a complete training cycle of a CNN; prediction performance of a CNN generally increases with the number of epochs as it makes correction to the decision weights over time, until improvements are no longer possible due to dataset limitations.
Steps per Epoch	Total number of training steps to be carried out in each epoch; usually determined by the total size of the dataset divided by the defined Batch Size.

5.2.2 Dataset Separation – Training and Validation

When setting out to train a CNN, it is important to work with a sufficiently large image dataset to ensure that the necessary features are extracted from the dataset for each

class to develop a robust predictive model. Additionally, the calculated weights associated with each filter for prediction applications must be tested against a validation dataset to ensure that the model can make accurate predictions on new images that were not part of the training set. A universally accepted and adopted breakdown of a dataset used for CNN model development is an 80-20 split for the training and validation sets, respectively.

Further modifications are also made to the training dataset, including operations such as rotation, translation, and resizing of the images. This serves to further augment the training dataset, to ensure that the model is trained on images with a large variety of orientations. This process helps facilitate the development of a CNN model that is not overfitted to specific orientations of the classification objects. An overfitted model performs poorly when it encounters input images that are significantly different from the training images used.

5.3 Performance Metrics

The goal of a CNN is to perform the necessary predictions on an input image based on the desired image processing task, as discussed at the beginning of this chapter. During the CNN training process, there are two key metrics that are used to characterize the performance of the model: accuracy and loss.

- Accuracy – this metric provides information on the number of correct predictions made by the model after each complete epoch. The accuracy of a model will provide its estimated performance in making a correct prediction when used with new input images.
- Loss – this metric provides information on the average error between the average calculated numerical values from the classification layers and the final output.

Some commonly used loss functions for image identification problems include the binary cross-entropy and categorical cross-entropy for binary and multi-class models, respectively.

5.4 Experimental Testing and Results

As mentioned previously in this chapter, the performance of a CNN can be heavily affected by the user-defined parameters presented in Section 5.2.1. To quantify the effects of these variables, a baseline configuration CNN was developed and tested. For the purposes of this test, max pooling layers are used for the CNN. The performance of this CNN will serve as the benchmark for comparison while each of the following parameters are varied individually:

Table 5.4: CNN model parameters to be modified

Modified CNN Model Parameters		
Filter Size	Number of Filters	Input Size
Pooling Size	Batch Size	

The performance metric for comparison in this section will be the prediction accuracy at the end of each epoch. This provides valuable information on the expected performance of the CNN model when it is used for prediction applications on new input images. As the training process involves the randomized use of images within the dataset, it is possible for the CNN to be subjected to outlier images that can skew the performance of the model. As such, all data presented for the prediction accuracy of each test case is the average of three separate training sets.

5.4.1 Training Dataset – Kaggle Cats and Dogs

For the experimental testing on the CNN training parameters discussed in this chapter, the Kaggle Cats and Dogs dataset is used. This dataset contains a $50 - 50$ split of 25,000 images of cats and dogs; it was designed for the training of a CNN to perform the task of image classification. The images in this dataset vary in size and have a low resolution for ease of processing. A randomized collection of sample images from the dataset can be found in Appendix A. As mentioned in Section 5.2.2, the dataset was randomly separated into both a training and validation pool, with an 80-20 split. These two sets of images for both cats and dogs are kept constant throughout the various test cases.

5.4.2 CNN Baseline Configuration and Performance

Table 5.5 contains the values used as the baseline configuration CNN that will serve as the benchmark. The bolded variables are maintained throughout all test cases. The average of the prediction accuracy after each epoch for three separate training runs of the CNN are presented in Figure 5.8.

Table 5.5: Baseline CNN configuration parameters

Filter Size	3×3
Number of Filters	32
Input Size	32×32
Max Pooling	2×2
Batch Size	16
Steps per Epoch	**20000/Batch Size**
Epoch	**10**

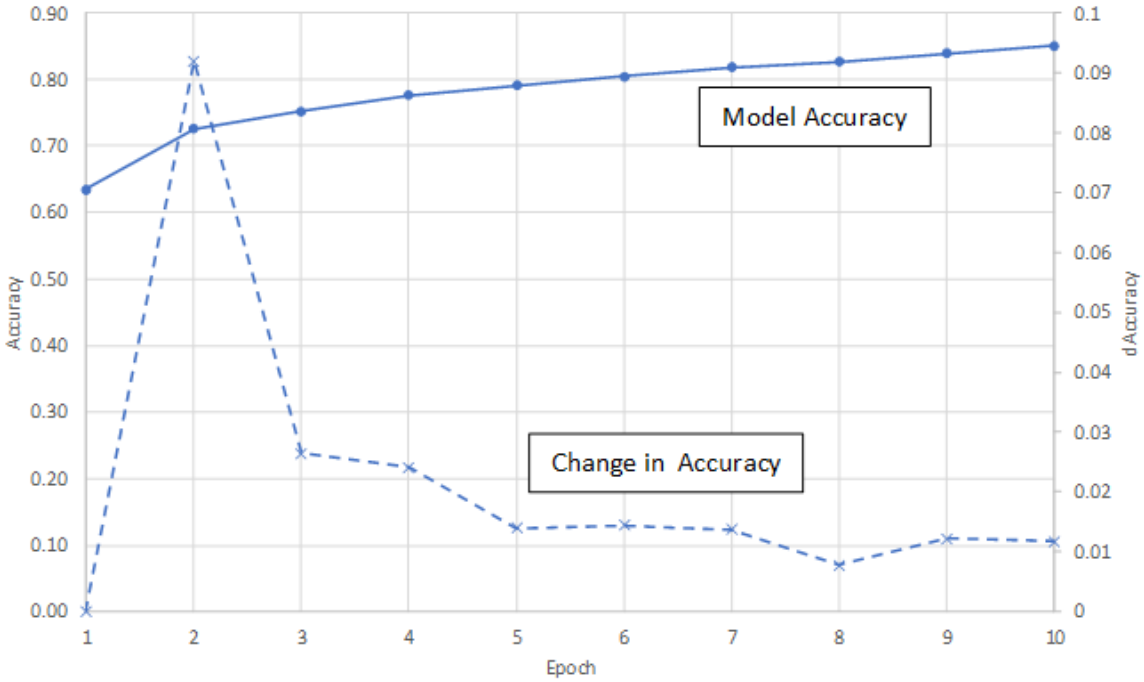

Figure 5.8: Baseline CNN prediction accuracy

The model prediction accuracy is represented by the solid blue line in Figure 5.8. The CNN achieves a 64.0% prediction accuracy after the first epoch and reaches a maximum of 85.4% after the tenth epoch. The dashed line displays the gain in prediction accuracy between epochs. Though it was not covered in the scope of this chapter, this can also be another metric for consideration when determining the total number of epochs for training; it can assist in maximizing the final prediction accuracy of the CNN model while minimizing the number of epochs for training.

5.4.3 Modification of User-Defined Parameters

Figure 5.9 displays the CNN prediction accuracy performed with the modifications made to the model parameters setup in Table 5.6. For each test case, the parameters are varied individually while maintaining the original CNN model configuration; the baseline configuration of the CNN can be found in Section 5.4.2.

Table 5.6: New CNN variables used for individual training sets

Filter Size	4×4
Number of Filters	64
Input Size	64×64
Max Pooling	4×4
Batch Size	32

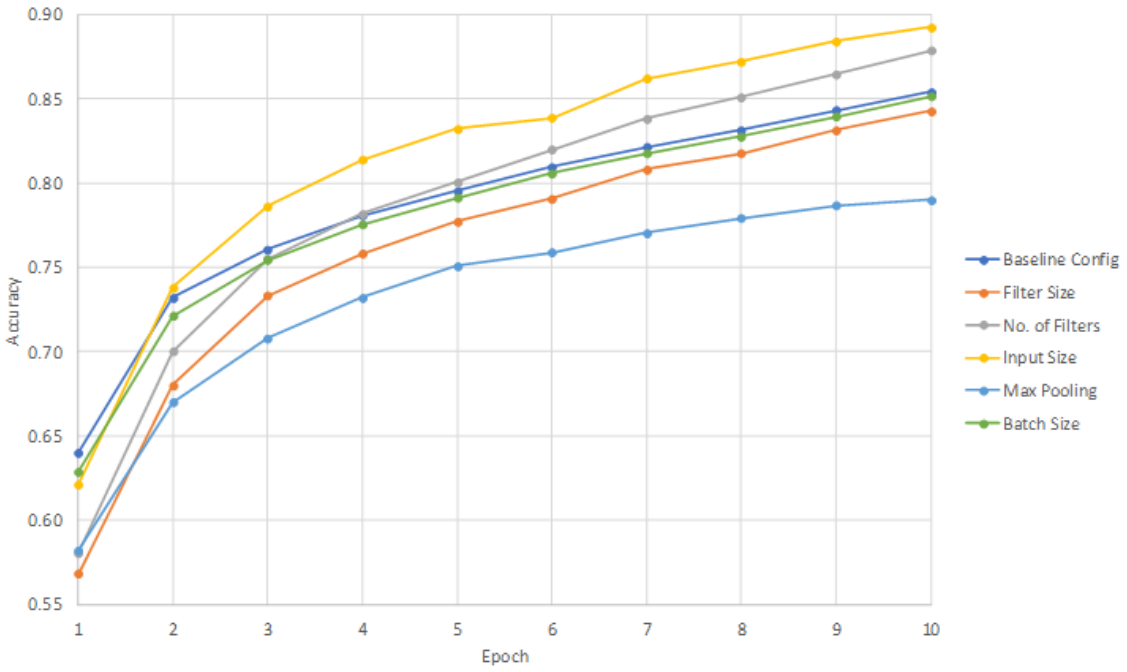

Figure 5.9: Comparison of baseline CNN performance with various test cases

The results from the test of the varied parameters in the CNN training present either a positive or negative gain in prediction accuracy. A similar curve trend exists between the each of the individual test cases; this is due to the learning process of a CNN, where there is a rapid increase in the first few epochs, and then change in accuracy between epochs begins to decrease. The effects of each of the varied parameters are summarized in Table 5.7, and a brief discussion is presented.

Table 5.7: Effects of varied CNN training parameters on prediction accuracy

Parameter	Type of Effect	Reasoning
Filter Size	Negative	By enlarging the filter size, larger characteristic features are identified in the convolution layers. This results in more generic characteristic features being identified. As such, the prediction accuracy of the model decreases due to more information being lost between layers during the convolution operation.
Number of Filters	Positive	By increasing the number of filters, more characteristic features are identified for each of the two possible outputs. This results in more filters that can be applied to a new image, which results in greater accuracy when making a prediction.
Input Size	Positive	By increasing the input size of the image, more information is available for the CNN to identify and locate characteristic features. This allows the CNN to identify more specific features that may be unique to either of the two output classes.
Max Pooling	Negative	By increasing the max pooling window in the pooling layer, there is a greater reduction in dimensionality of the data. This loss of data results in a lower prediction accuracy, as the developed filters in the ensuing convolutional layers have less data to work with.
Batch Size	Negligible	By increasing the batch size, the processing time of each training step in an epoch increases. However, as the dataset used is ultimately the same, it has a negligible effect on the prediction accuracy of the final CNN model.

5.4.4 Proposed CNN Model Setup

With the information learned from the comparative study performed in Section 5.4.3, a new model configuration for the CNN to leverage a net positive gain in prediction accuracy was developed. The 'Number of Filters' and 'Input Size' were updated to the values used in Section 5.4.3, while maintaining the remaining variables according to the baseline CNN training configuration. To compare the performance of this new CNN training parameter configuration, results from the baseline setup as well as the test cases where a positive increase in prediction accuracy occurred are used for comparison; the results are presented in Figure 5.10.

The increased number of filters and the larger input image size both enable the CNN to identify more characteristic features during each training epoch. There is an overall increase to the starting and ending model prediction accuracy when compared to the baseline CNN model. Another key observation is the lower prediction accuracy in the first few epochs of the training set when compared to the "Input Size" test case; this is likely due to the increased number of filters that affects the initial prediction accuracy of the model. With a larger number of filters, more characteristic features are identified during the feature extraction layers of the CNN; as some of these features may not be unique to a specific class, more time is needed to further adjust the weights associated with each filter in the classification layers to make a prediction on the object class. The modified configuration CNN achieved a final prediction accuracy of 91.9% after the tenth epoch.

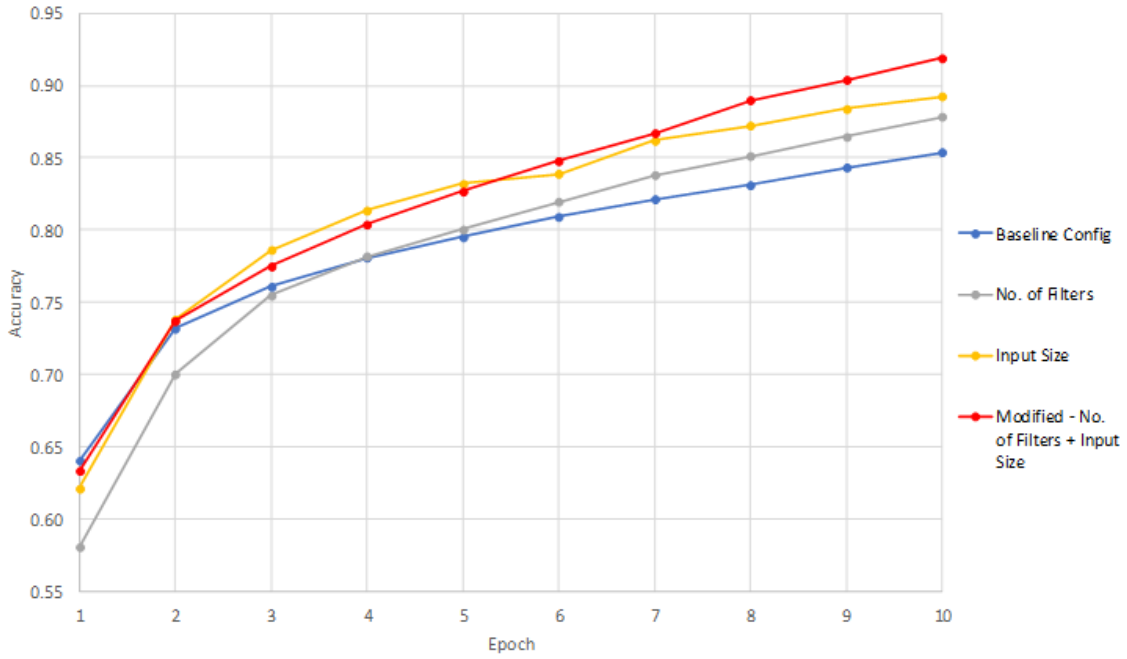

Figure 5.10: Comparison of modified CNN prediction accuracy against baseline and select test cases

5.5 Conclusions

This chapter summarized the operating theory of a CNN for image processing applications and studied the effects of specific user-defined parameters associated with the development of the feature extraction portion of a CNN model. Individual test cases were conducted on each parameter separately to characterize the effects of each on the overall performance of the CNN with respect to the final prediction accuracy. A further modified set of parameters that was built upon this information was then used to retrain the CNN; this test case produced the highest prediction accuracy after ten epochs.

Chapter 6: Development of a Vision-Based Positioning System for Applications on a Remotely Piloted Aircraft

Many modern RPA come equipped with an image sensor; this is due to the copious amounts of data that can be mined from the visual imagery collected onboard during flight. As discussed in the previous chapters of this thesis, computer vision techniques can be applied to the imagery data collected to process and identify key information, such as localization and classification of targets within the image frame. The benefits of the image processing methods presented in the previous chapters include being able to either post-process the visual information after a flight, or also to be able to use the information collected in real-time for improved RPA operations.

Satellite-based localization is one of the most widely used systems onboard modern-day RPA. This method of localization provides a good combination of positional accuracy in the horizontal ($\pm$ 1.9 m) and the vertical ($\pm$ 3.9 m) directions (Applied Research Laboratories, 2019), and the ability to provide geographical coordinates and altitude for waypoint following missions for RPAS operations. The development of a VPS can be beneficial to RPAS, as it can be used as an additional method of localization, to further enhance positional accuracy. In some cases, it can also serve as the main source of information for localization in situations where satellite-based systems are inoperable or unavailable (e.g. due to jamming or during underground operations).

This chapter outlines the process of developing a VPS that can be used to provide localization information to an aircraft based on a known target. By utilizing the computer vision framework presented in Chapter 3, it is possible to develop a highly customizable VPS that can be suited to its specific application.

6.1 3D Computer Vision – 2D Image Coordinates to 3D World Coordinates

When an image is taken, the position information of all objects within the image frame are stored in the 2-dimensional pixel coordinate frame. With the application of computer vision techniques, it is possible to convert that image coordinate to its position in the 3-dimensional world coordinate frame. This information can then be used to determine the position of the image sensor, and by extension, the RPA that it is equipped on for localization applications.

For 3D computer vision applications, there are three main coordinate frames that must be considered; utilizing known parameters of the image sensor, the 3D position information of a point can be converted between these three coordinate frames. Figure 6.1 provides an illustration of the three coordinate frames – the image coordinate frame, the camera coordinate frame, and the world coordinate frame.

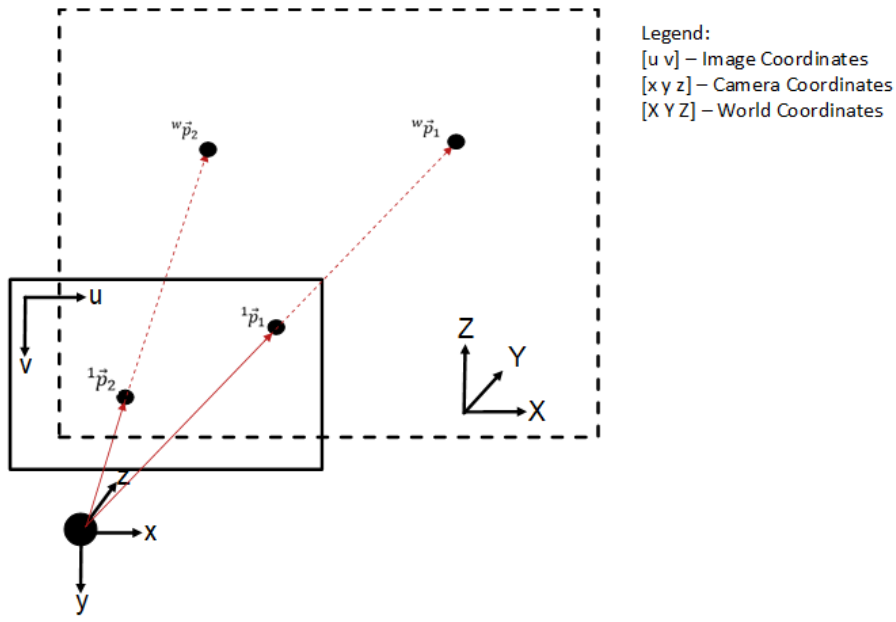

Figure 6.1: Definition of a 3D point in three different coordinate frames

To perform the conversion between these coordinate frames, further information is required on the image sensor's intrinsic and extrinsic parameters. The intrinsic parameters

of an image sensor include information on its focal length, optical center, and skew coefficient. These parameters are stored in the form of a matrix, $\boldsymbol{K}$, and is commonly referred to as the intrinsic parameter matrix.

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Where:

$f_x, f_y = Image\ sensor\ focal\ lengths$

$c_x, c_y = Image\ sensor\ optical\ centers$

$s = Skew\ coefficient$

The extrinsic parameters of an image sensor relate to its 3D world position with respect to the 3D point of interest. The extrinsic parameter matrix, $\boldsymbol{M}$, is made up of two components that are derived from the pose of the image sensor – the translation vector, $\vec{T}$ and the rotation matrix, $\boldsymbol{R}$. The rotation matrix used in the M matrix is based on the necessary rotations required to have the camera coordinate frame match the orientation of the world coordinate frame.

$$\boldsymbol{M} = \left[\boldsymbol{R} \mid \vec{T}\right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where:

$r_{11}, r_{12}, r_{13} = Rotation\ vector\ about\ X\ axis$

$r_{21}, r_{22}, r_{23} = Rotation\ vector\ about\ Y\ axis$

$r_{31}, r_{32}, r_{33} = Rotation\ vector\ about\ Z\ axis$

$t_x, t_y, t_z = Translation\ vector$

To convert between the three reference coordinate frames, the following relationships shown in Equation 6.1 and Equation 6.2 can be used:

Image Coordinate Frame – Camera coordinate frame:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{(Eq 6.1)}$$

Camera Coordinate Frame - World Coordinate Frame:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \text{(Eq 6.2)}$$

The orientation of the coordinate frames displayed in Figure 6.2 is shown as an example. When designing a VPS to determine the position of a tracked target, the system designer can define each of the coordinate frames as necessary; the 3D position information can be determined regardless of the orientation of the various coordinate frames, provided that it is defined and kept constant throughout the coordinate conversion process. For the purposes of the VPS discussed in this chapter, Figure 6.2 represents the orientation of the three separate coordinate frames.

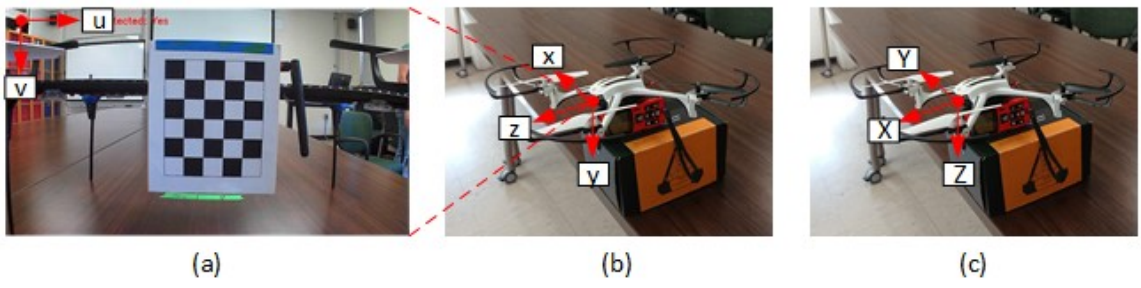

Figure 6.2: Coordinate frame definition for (a) image coordinate frame (b) camera coordinate frame (c) world coordinate frame

6.2 Camera Calibration

When looking to use an image sensor for 3D machine vision applications, the first step that must be taken is to perform camera calibration. Camera calibration is important, as it helps identify key parameters related to the intrinsic and extrinsic properties of the sensor necessary to perform the conversion between coordinate frames, as discussed in Section 6.1. The camera calibration process revolves around the use of a target of known dimensions being recorded in different orientations; a commonly used target for this application is a checkerboard pattern. This is due to the pattern forming a grid that can be easily identified by computer vision algorithms, and its real world coordinates are simplified to just the X and Y coordinates, as its Z coordinate value is constant throughout the grid as it lies on a flat plane.

The image processing library used for this work, OpenCV, has built-in functions to easily facilitate the process of conducting image sensor calibration. The cv.findChessboardCorners() function automatically searches the image frame to identify a chessboard of size m rows by n columns and highlights the corners. For example, the chessboard pattern shown in Figure 6.3 is of size 5×7, as the search algorithm searches for the internal corners of the pattern. This information can then be correlated to their respective 3D world coordinates. When the pixel coordinates are identified, it is possible to store them in one of two units – either as a unit length of the square edge or as the real-world distances. A test was conducted on both methods, and the use of the real-world distances resulted in very noisy position estimations. As such, the recorded information for the camera calibration and position estimations are left in the unit lengths of the chessboard square edge, which can then be converted to the real-world measurements after.

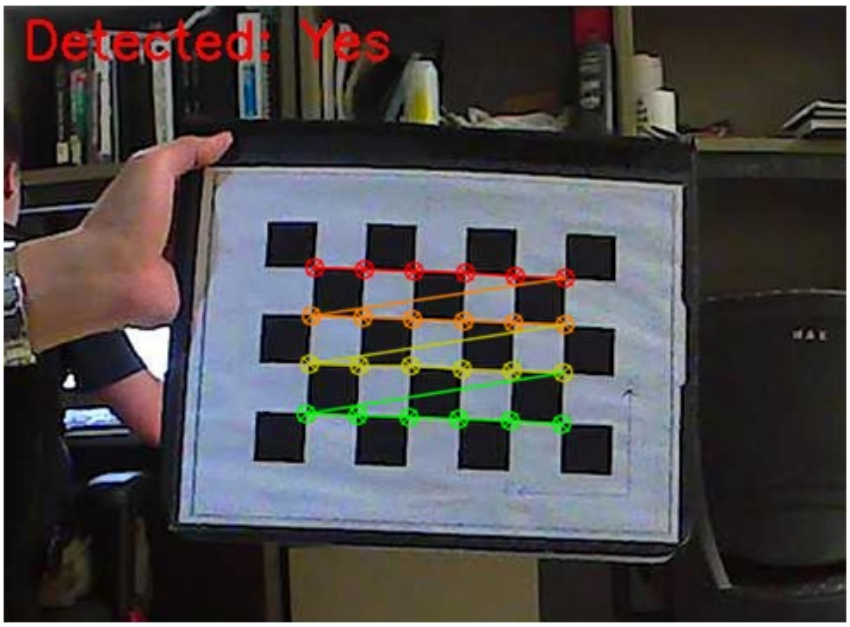

Figure 6.3: Illustration of intersection point detection by cv.findChessboardCorners()

For best results, a recommended minimum of 10 input images of the chessboard target in randomized orientations is needed to perform the camera calibration step. The cv.calibrateCamera() function is used, which correlates the world coordinates of each of the highlighted corners with its associated pixel coordinates. By doing so, the function is then able to identify the image sensor intrinsic parameter matrix, $\boldsymbol{K}$. Using this K matrix, the cv.solvePnP() function can be used to determine the pose of the target; this includes both the translation vector, $\vec{T}$ and rotation vectors required to form the rotation matrix, $\boldsymbol{R}$.

6.3 Experimental Testing and Results

For this chapter, the experimental work conducted was carried out in multiple phases; the first phase involved developing a suitable camera calibration image collection procedure to obtain the best results for estimation of translation. The second phase involved the testing of the capabilities of the proposed VPS to perform real-time processing of tracking a known target and computing the associated output.

6.3.1 Phase 1 – Camera Calibration Methods

To assess the performance of the cv.solvePnP() function in determining $\vec{T}$ and $\boldsymbol{R}$, three separate calibration image sets were taken. Each of the three calibration image sets were collected with differing parameters to identify the best method of obtaining the ; these are summarized in Table 6.1. Each of these calibration image sets are then used for camera calibration using the cv.calibrateCamera() function.

Table 6.1: Calibration image set collection parameters

Calibration Set	Total Number of Images	Target Location
1	20	Randomized distances and orientations; up to 150 cm away from the image sensor.
2	25	5 randomized orientations each at 20 cm intervals away from the image sensor, starting at 20 cm and ending at 100 cm.
3	21	7 randomized orientations each at 50 cm intervals away from the image sensor, starting at 50 cm and ending at 150 cm.

The resulting $\boldsymbol{K}$ matrix obtained from each of the three calibration image sets are then used as the inputs to the cv.solvePnP() function to identify the $\vec{T}$ vector and $\boldsymbol{R}$ matrix of the target. Although the VPS can provide an estimation for both the translation and rotation of the target, the experimental work presented in this chapter will focus only on the translation estimation. For this test, a chessboard pattern of dimensions 6×5 and corner lengths of 1.5 cm is attached to the back of a 3DR Iris+ to simulate tracking of an RPAS equipped with the target; the experimental setup is shown in Figure 6.4. A static test

was conducted, with the target being moved away from the image sensor; 5 measurements each from the VPS of the estimated distance of the target were taken at 20 cm intervals.

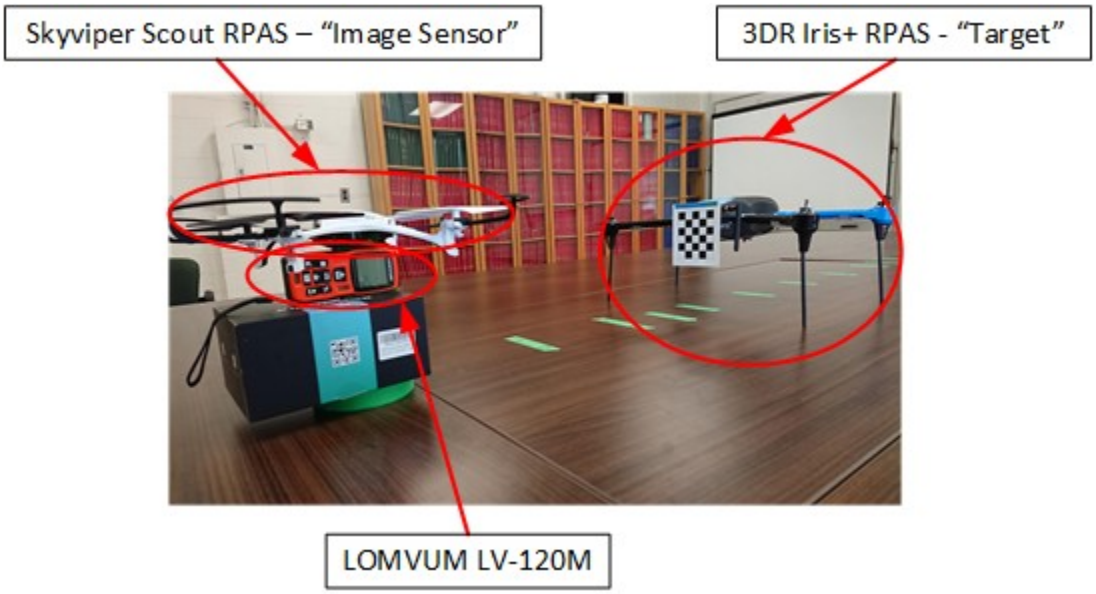

Figure 6.4: Experimental setup for camera calibration tests

6.3.1.1 Calibration Set 1: Randomized Locations

Table 6.2: Distance measurements comparison for calibration set 1

Test Point	LDM Reading (mm)	VPS Reading (Average) (mm)	Percent Difference (%)
1	207	223	7.69
2	428	442	3.20
3	610	625	2.53
4	813	830	2.04
5	1009	1077	6.74
6	1214	1575	29.71
7	1404	1774	26.34

The results of this camera calibration test provide good position estimates at distances up to approximately 1 m away from the image sensor as given in Table 6.2. The

VPS measurement difference with the LDM range between a low of 1.4 cm (test point 2) and a high of 6.8 cm (test point 5). However, performance drops significantly beyond the 1 m mark, where the percent difference increases to approximately 30%. This sudden drop in performance is likely due to the low resolution of the image sensor, which may have exceeded the threshold of resolution required for the VPS algorithm to accurately identify the corners of the chessboard squares. The noisy data measurements can be seen in the tabulation of data for the separate readings taken at each test point in Appendix B.

6.3.1.2 Calibration Set 2: 20 cm Intervals

Table 6.3: Distance measurements comparison for calibration set 2

Test Point	LDM Reading (mm)	VPS Reading (Average) (mm)	Percent Difference (%)
1	214	148	31.05
2	421	287	31.79
3	619	424	31.42
4	816	558	31.59
5	1012	712	29.60
6	1213	1047	13.70
7	1414	1195	15.48

This calibration test produced the least accurate results, with an average percent difference in measurement of 26.29%. Upon review of the calibration image set collected, a majority of the images at each of the five distance intervals were placed at large offset angles in relation to the image plane. This may have led to the cv.calibrateCamera() function to produce a bad estimation for the intrinsic and extrinsic parameter matrices,

which would result in the inaccurate distance estimation seen in the results of this calibration set.

6.3.1.3 Calibration Set 3: 50 cm Intervals

Table 6.4: Distance measurements comparison for calibration set 3

Test Point	LDM Reading (mm)	VPS Reading (Average) (mm)	Percent Difference (%)
1	220	191	13.27
2	425	345	18.86
3	627	505	19.39
4	820	658	19.80
5	1017	852	16.21
6	1216	1292	6.23
7	1415	1454	2.74

Table 6.4 shows the results of the distance measurement comparison for the third calibration test. This image set resulted in better performance than the previous set but is overall still less accurate than the results of the first calibration set. There is a significant improvement on the distance estimation at further distances, beyond the 1 m range where calibration set 1 excelled. The cause of this sudden improvement may be due to the pose of the target in the calibration image set; upon review of the image set, the target was mostly perpendicular to the image frame. This would enable the cv.findChessboardCorners() to more accurately locate the corners, thus providing better accuracy results.

With the results obtained from each separate test, the best method for collecting a camera calibration image set would be to ensure that the target is captured at randomized

locations and orientations within the image frame. A mix of target orientations, ranging from perpendicular to large angles, must be achieved to produce a good camera calibration image set. Additionally, to address the relatively short range of measurement, the target size is increased for the next phase of tests conducted with the VPS.

6.3.2 Phase 2 – Development of Post-Processing Methodology

The next series of tests conducted on the VPS was to determine its ability in measuring distances in all three translational directions. For this set of tests, the VPS script was executed and the video feed from the Sky Viper Journey RPA was processed in real-time on the GCS laptop. The information recorded for these tests include the VPS position estimation and the time taken to perform the computation for each measurement taken. A sample section of the results table from one test set is included in Appendix B. Additionally, to further assess the capabilities of the VPS, the target template was increased to be of dimensions 7×5, with sides of length 2.8 cm.

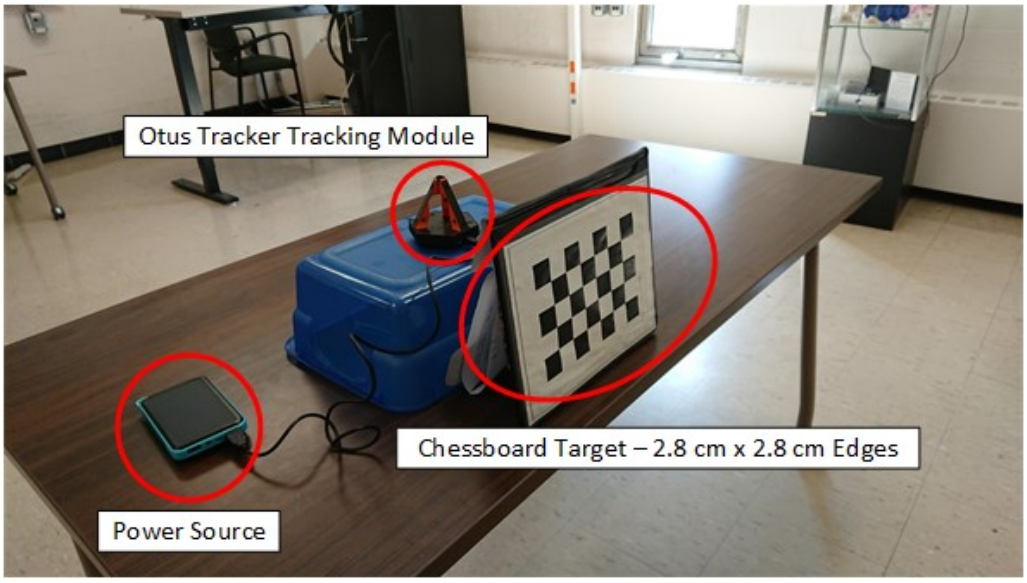

Figure 6.5: Target setup for VPS and Otus Tracker position tracking

The Otus Tracker system discussed in Chapter 3 is used as the measurement apparatus for comparison, as the output of the system can be compared directly with that of the VPS. Initially, the tracking of displacement in all three axes are individually tested, and a final test was conducted with free motion of the target in 3D space. To accommodate for the use of a larger target for these sets of tests, a new set of 24 calibration images were taken with the lessons learned from Section 6.3.1 in mind.

6.3.2.1 Translation Test 1 – X-Axis

For the translation tests conducted in the X-axis, a total of five separate trials were conducted. These tests were started at approximately 50 cm away from the image sensor, and the target is incrementally moved away, with a maximum distance of 250 cm. These tests allowed for real-time benchmarking of the proposed VPS, allowing for the determination of the average update rate of the position information that can be provided.

Figure 6.6 displays the comparison of the position measurements obtained by the VPS and the Otus Tracker system in the X-axis of the world coordinate frame from trial 3 of the experimental test set. The VPS provides comparable measurements on the displacement of the target to the Otus Tracker. The VPS begins to experience significant degradation in position estimation at approximately 190 cm away from the image sensor. Similar to the sharp decrease in accuracy seen in the results presented in Table 6.2, this is likely due to the loss of information in terms of image resolution; the search algorithm is unable accurately identify the edges of the chessboard to perform the necessary computation to produce an accurate position estimate.

As the VPS has a variable position update rate that is dependent upon the time taken to compute the position of the target, a direct comparison of the difference in position

between the VPS and the Otus Tracker cannot be computed. The data collected from both systems are processed in Matlab, where both data sets were linearly interpolated to provide an estimated position at each of the data points that exist for both systems. This method of interpolation does not necessarily provide an accurate representation of the estimated position from each system, as it linearly interpolates the position estimate between two data points based on the time scale. However, as this system was custom designed, there is no pre-existing model from which a prediction function could be derived for better interpolation of the data points.

Table 6.5: VPS performance metrics for trial 3

Max X-Axis Measurement Difference (cm)	68
Average X-Axis Measurement Difference (cm)	9
Max Update Rate (Hz)	39
Min Update Rate (Hz)	3
Average Update Rate (Hz)	11

Table 6.5 provides a tabulation of the VPS system performance, including the maximum and average position measurement differences between itself and the Otus Tracker. The larger average measurement difference most likely spans from the operation of the VPS beyond its optimal position estimation range, as seen in between the time range of $t = 30s$ and $t = 60s$ in Figure 6.6. Additionally, there is a period between $t = 70s$ and $t = 75s$ where the Otus Tracker produces an inaccurate negative reading after the movement of the target was halted.

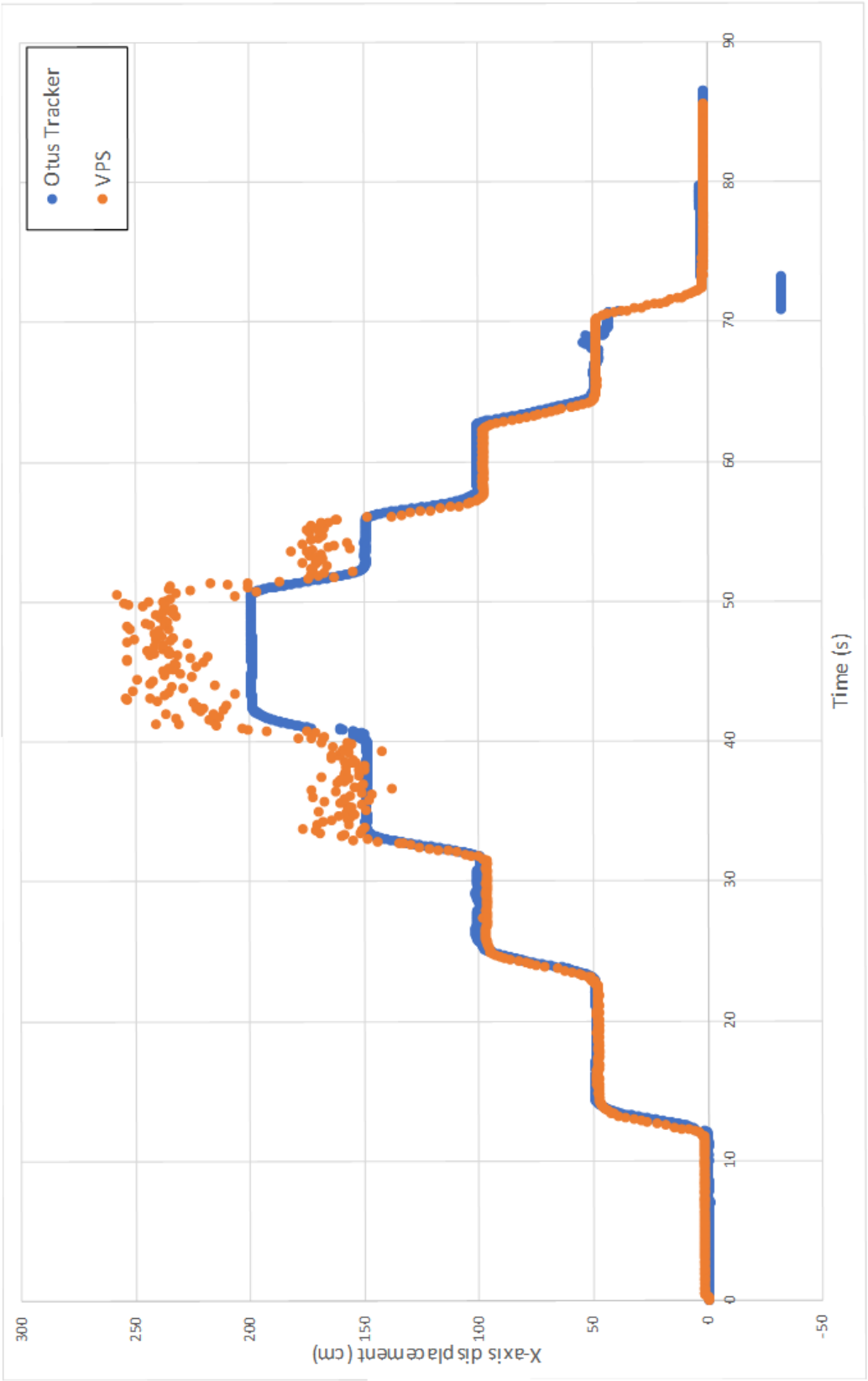

Figure 6.6: Comparison of position estimation in X-axis for trial 3

6.3.2.2 Translation Tests 2 and 3 – Y-Axis and Z-Axis

For the tests conducted to determine the performance of the VPS in tracking the target's position in the Y-axis and Z-axis, a total of four separate tests were conducted for each. These tests were conducted at preset distances away from the image sensor, ranging between 50 cm to 200 cm away from the image sensor, with a distance interval of 50 cm. Figure 6.7 is an example setup for the tests conducted for the Y-axis.

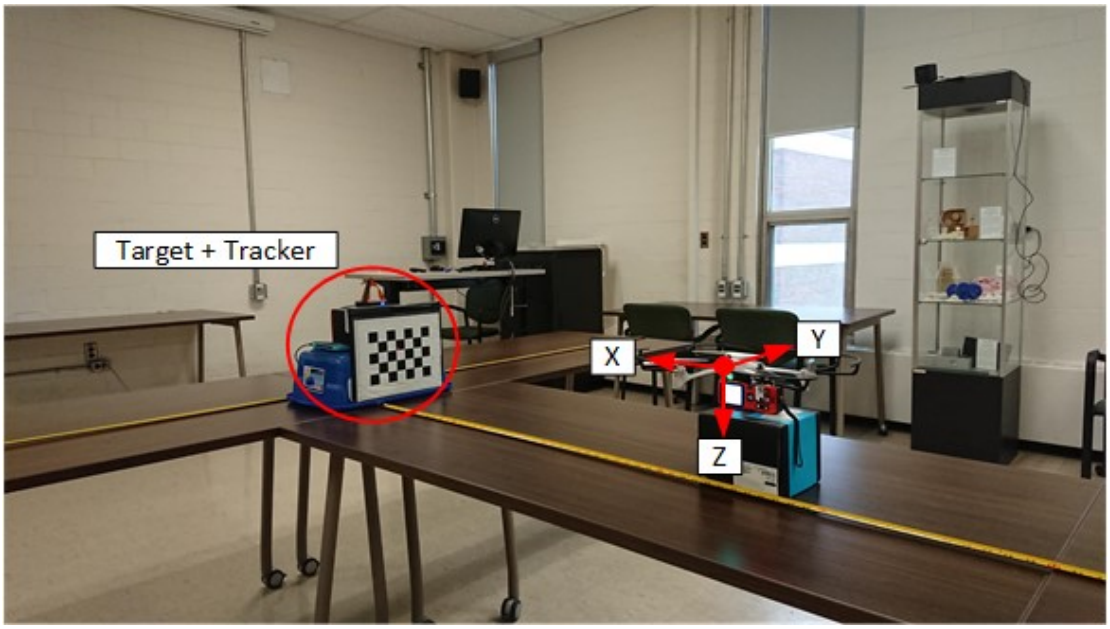

Figure 6.7: Experimental setup for translation tests in Y and Z-axis

The performance metrics of the tests conducted for both the Y and Z-axis at X = 150 cm are tabulated in Table 6.6 and Table 6.7, respectively.

Table 6.6: VPS performance metrics for Y-axis @ X = 150 cm

Max Y-axis Measurement Difference (cm)	27
Average Y-axis Measurement Difference (cm)	8
Max Update Rate (Hz)	42
Min Update Rate (Hz)	1
Average Update Rate (Hz)	12

Table 6.7: VPS performance metrics for Z-axis @ X = 150 cm

Max Z-axis Measurement Difference (cm)	12
Average Z-axis Measurement Difference (cm)	4
Max Update Rate (Hz)	40
Min Update Rate (Hz)	1
Average Update Rate (Hz)	12

Figure 6.8 and Figure 6.9 are the plots of the tracked displacement of the target in the Y and Z-axis tests conducted at X = 150 cm. Unlike the results obtained from the test conducted in the X-axis, there is a noticeable discrepancy between the measurements provided by the VPS and the Otus Tracker. This situation occurs due to the optical distortion caused by the lens of the image sensor. Lens distortion affects the representation of an object in the image frame; it may cause straight lines to appear curved due to the geometry of the lens. For certain imaging systems, the lens distortion is accounted for and corrected prior to the image being processed. However, for lower cost image sensor such as the one found on the Sky Viper Scout, lens distortion is not corrected for due to its limited computation power.

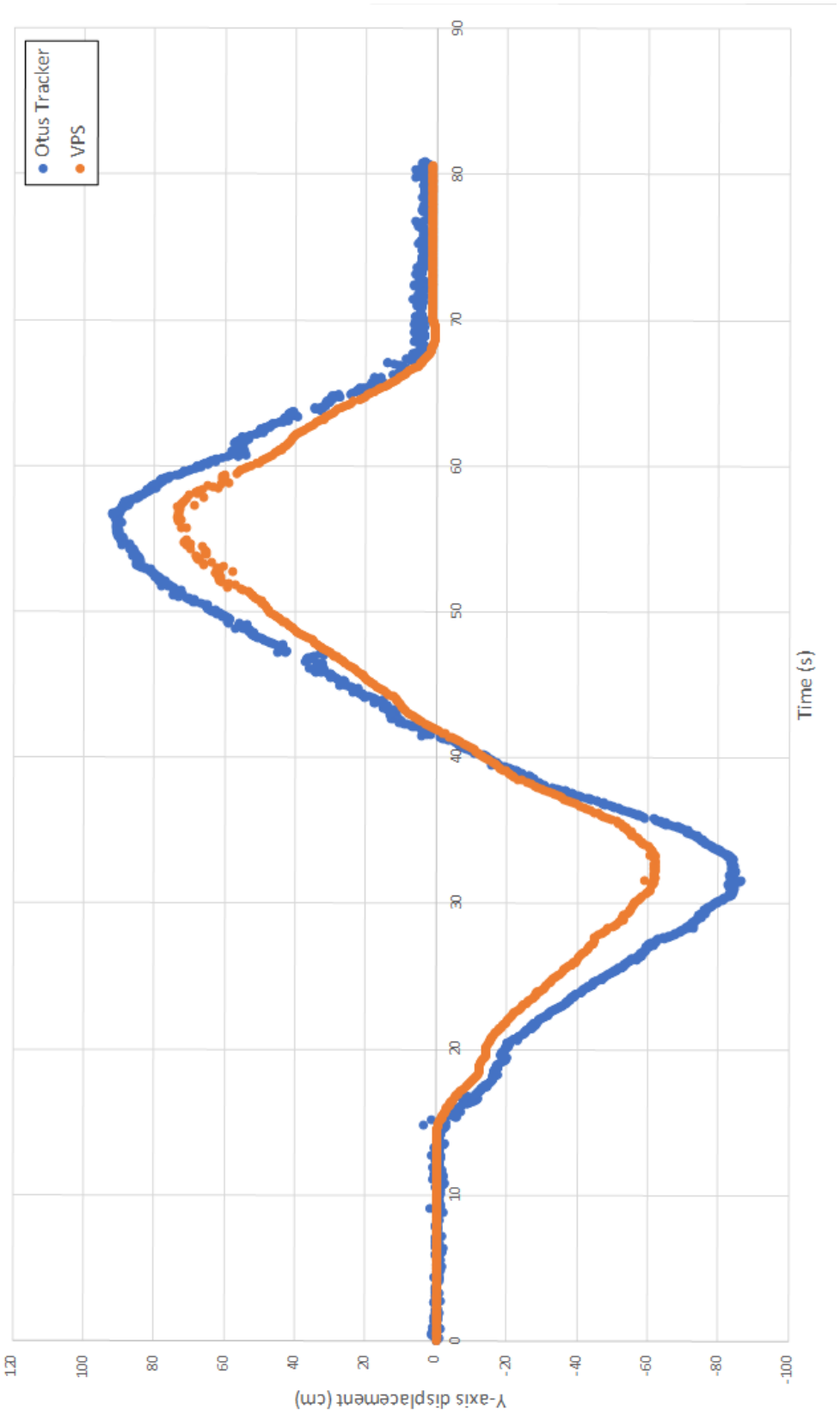

Figure 6.8: Comparison of position estimation in Y-axis @ X = 150 cm

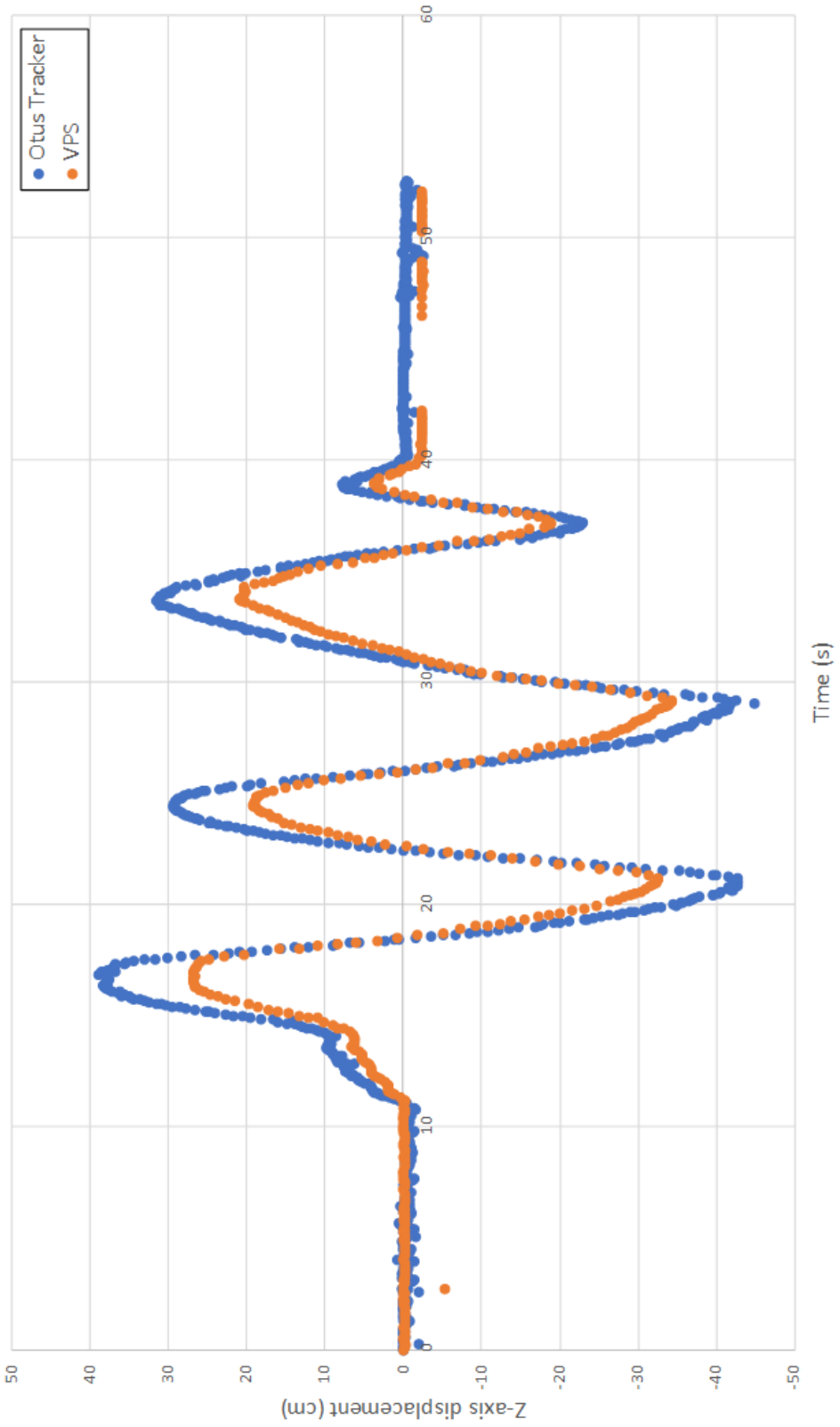

Figure 6.9: Comparison of position estimation in Z-axis @ X = 150 cm

6.3.2.3 Characterization of Vision-based Positioning System Operational Range

Further analysis work was conducted on the position data collected in the X-axis test presented in Section 6.3.2.1. As seen from Figure 6.6, the VPS reports position information on the target that coincides with that of the Otus Tracker up to approximately 190 cm (the starting point of the target was 50 cm away from the image sensor). Beyond this point, the difference in position measurement increases with the distance of the target from the image sensor; the VPS reports position estimation values that are greater than the measurements from the Otus Tracker.

The measurement difference between both the VPS and the Otus Tracker across the collected dataset was calculated and is presented in Figure 6.10. This plot displays the relationship between the measurement difference reported by the VPS and the distance of the target from the image sensor. The average percent difference between measurement readings were calculated at 50 cm intervals and are summarized in Table 6.8.

Table 6.8: VPS measurement average percent difference at 50 cm intervals for X-axis trial 3

X Position (cm)	Average Percent Difference (%)
50	2
100	3
150	9
200	17

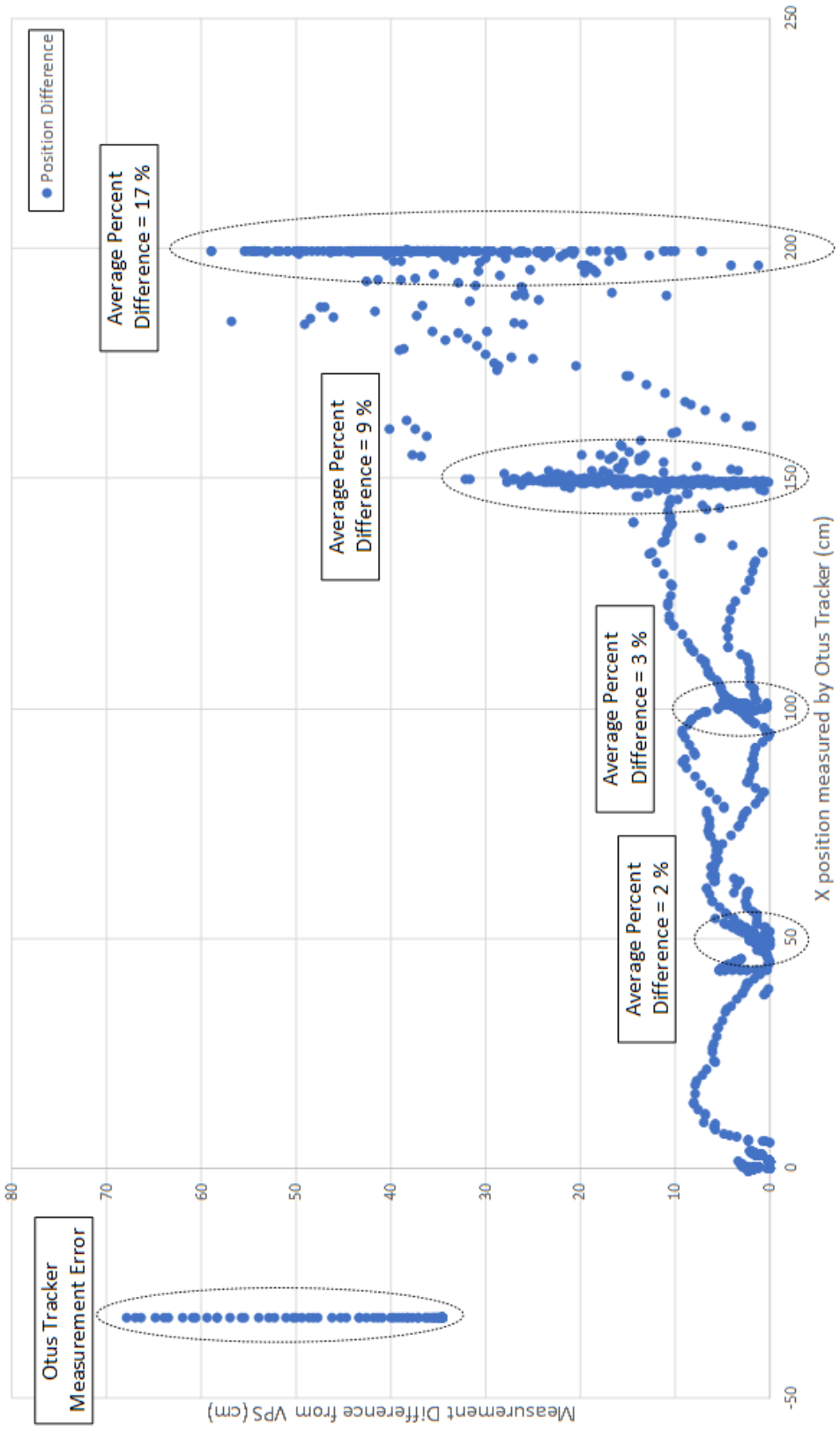

Figure 6.10: Position difference between Otus Tracker and VPS – X-axis trial 3

The performance of the VPS can be characterized by identifying the average percent difference in position measurements between the VPS and the Otus Tracker. The results tabulated in Table 6.8 can be used to predict the position measurement accuracy of the VPS at various distances – this information can be used in the post-processing step to determine the potential measurement errors provided by the VPS based on the distance of the target from the image sensor.

6.4 Overall Performance

Table 6.9: Performance of VPS across all tests

	Number of Tests	Average Measurement Difference (cm)	Average Update Rate (Hz)
X–axis	5	34	12
Y-axis	4	4	12
Z-axis	4	11	11
Overall	13	18	12

Table 6.9 provides a breakdown of the various tests conducted for phase 2 of this work, with the average performance of the system being determined. The large average measurement distance seen in the X-axis tests is due to the distance measurement being taken up until the point where the VPS begins to provide inaccurate position estimates. Additionally, there were certain regions during the experiment where the Otus tracker was unable to track the target and provided incorrect readings.

6.5 Conclusions

This chapter discussed the development process of a VPS that can be used for localization applications onboard an RPA. For the work presented in phase 1, different camera calibration image sets were assembled, and the results of the accuracy in

measurement resulting from each dataset is presented. Utilizing the information and lessons learned from the camera calibration process, a new calibration image set was constructed for use with a larger target to enable greater distance measurements with the VPS. Phase 2 of the work involved testing the VPS to determine the level of accuracy that can be achieved for all three translation axes of the aircraft in the 3D world coordinate frame.

Chapter 7: Conclusions and Future Recommendations

This thesis presents a computer vision framework for improved RPA operations. As image sensors are rapidly becoming a common payload found onboard consumer-level RPA, it is possible to leverage the presence of these sensors to maximize the potential of these systems. By breaking down the image analysis task into the categories presented in the computer vision framework, an image sensor can be assessed to ensure that it can provide the necessary visual imagery data to enable completion of its mission-specific task.

The process outlined in Chapter 4 was developed to assist RPA operators in assessing the capabilities of an imaging system, and to enable suitable sensor selection to meet data quality needs. By determining the *GSD* of an image sensor and using the Johnson criteria, it is possible to predict the outcome of a trained human observer in performing each of the three image classification tasks discussed. This information can be used to develop the mission plan to ensure that the image sensor is able to provide the necessary data quality to meet classification task requirements.

The CNN development work presented in Chapter 5 was geared towards developing a better understanding of the concepts behind CNN modelling. The two parameters that positively affected the CNN performance is the “Input Size” and “No. of Filters” – both of which revolve around the processing of a greater volume of data. By increasing the size of the input image and the number of filters to be identified during training, a CNN can more accurately predict the object class located within the image.

The experimental work presented in Chapter 6 serves as a starting point for the development of a VPS that can be used for RPA localization and precision maneuvers. The system achieves an average measurement difference of 18 cm when compared to a

commercially available localization system. The average measurement update rate of the system is 12 Hz.

7.1 Summary of Contributions

The major results and findings presented in this thesis can be classified into two categories: generalized and specific. Generalized results and findings refer to information discovered that correlates to the general application of the technologies presented towards RPA systems and operations. Specialized results and findings, on the other hand, are tied to the specifications of the equipment used throughout the experimental work presented.

Generalized contributions:

- Developed a computer vision framework to enable improved RPA operations.
- Developed guidelines for analyzing an image sensor to determine its capability to meet the necessary image quality level required for a human observer to perform the image classification task of detection, recognition, and identification.
- Provided a high-level overview of CNNs for image processing applications and identified commonly varied CNN model parameters that can be manipulated through Keras. The availability of more visual data during the training process leads to greater model prediction accuracy, however it does come with a penalty towards the overall model training time.
- Experimentally tested the capabilities of a VPS in providing position information for RPA localization applications in both real-time and post-processing settings. Identification of position data analysis methods to assist in characterization of VPS operational limitations.

Specific contributions:

- Developed process for the utilization of the Johnson criteria and image sensor specifications to develop an operational envelope of RPA to meet desired image quality levels with the DJI Mavic 2 Pro.
- Identified "No. of Filters" and "Input Size" as two CNN model parameters that can positively affect the final model prediction accuracy for image analysis applications. The modified configuration of the CNN model achieved a final model prediction accuracy of 91.9%.
- Developed a VPS utilizing an off-the-shelf RPA with an onboard image sensor and a mid-range laptop that can serve as a GCS. This specific setup of the VPS can be used for RPA localization during the post-processing step of imagery data analysis that reports an average position measurement difference of 18 cm when compared to a commercially available indoor localization system. This VPS also has a position update rate of 12 Hz, which has an update rate that is greater than standard satellite-based systems.

7.2 Future Research

This section of the chapter will cover discussions on future work that can be conducted to further develop and improve upon the findings and results presented in this thesis. The following three subsections below are directly correlated to chapters 4 through 6 of this thesis, respectively.

7.2.1 Image Sensor Qualitative Study

With the large volume of imagery data that can be collected by an RPA in a single flight, the image classification task can be tedious for a human observer to sort through. As

such, from an image processing perspective, this classification process can seek to leverage the capabilities of a CNN to perform this task. However, the number of cycles necessary for a CNN to make an accurate prediction for each of the three classification tasks may differ from that of a human observer. Future work that can be expanded from this chapter would be to conduct the same image sensor quality study with known targets for classification in the image frame. This information can then be correlated with the outputs of a CNN for image classification, and the appropriate number of cycles can be determined for each task accordingly. Additionally, the development of the probability maps presented in Chapter 4 can be used to improve RPA operations; by knowing the flight parameters needed to meet data quality requirements, the ground coverage for each flight can be maximized.

7.2.2 Image Processing using a Convolutional Neural Network

A greater focus was placed on the convolution and pooling layers, as these provide great amounts of flexibility in the development of a CNN model. Future possibilities for improvement on the CNN model would be to allow it to continue training for more epochs; the modified CNN model presented in Section 5.4.4 still exhibited a large slope in its curve at the tenth epoch. The CNN training process can be extended further to take full advantage of the learning capabilities of a CNN until a plateau in prediction accuracy is reached. A metric for achieving a good balance between maximizing prediction accuracy and minimizing number of training epochs would be to calculate the change in accuracy between epochs and stopping the training process when the difference drops below a preset threshold.

The application of the work presented in this chapter is also targeted towards the development of a CNN to identify objects that are not classified by the pre-trained models discussed in Section 2.2; those models have been rigorously tested and retrained to ensure good prediction accuracy for general applications. The concept of transfer learning (Brownlee, 2019) should be visited for classification of uncommon objects; this will expand on the existing capabilities of the pre-trained CNN and reduces the time needed to train a model. This can then be compared against the development of a new CNN, which may be time consuming. For application specific CNNs, a suitably large dataset must be acquired. As such, another future work for CNN development should include a study to determine a suitable number of images necessary for each object class to achieve desired prediction accuracy performance.

7.2.3 Development of a Vision-based Positioning System for Applications on a Remotely Piloted Aircraft

The existing VPS can be used for post-processing applications; by attaching the target to another aircraft, its position in relation to the VPS can be determined if it is captured within the image frame. Future work for this chapter includes further testing and development of the VPS to minimize measurement errors to enable better tracking of the target. For example, the distortion caused by the lens on the image sensor used can be corrected to increase the accuracy of the measurement estimation produced by the VPS. The information provided by the current iteration of the VPS includes the output of the translation information of the system in relation to the target; the same cv.solvePnP() function is also able to provide an estimate for the rotation of the image sensor. This can be used in partnership with the translation information to develop target tracking and

following algorithms; it can also be integrated with other image analysis algorithms to enable further task-automation, such as collision avoidance and precision landing.

Finally, the VPS should also be ported over to a companion computer, such as the Raspberry Pi 3B+ that was discussed in Section 3.5.2 for integration to a small RPA. Testing should be done to benchmark the performance of the VPS on a companion computer that has a lower computing power, and further optimization of the image processing script should be done to maximize real-time performance onboard the RPA under these constraints. This will minimize the possibility of the VPS being inoperable due to an unstable data transfer connection or operations in a large area beyond the reach of existing data transfer connection capabilities.

Bibliography

Applied Research Laboratories. (2019). *An Analysis of Global Positioning System (GPS) Standard Positioning Service (SPS) Performance for 2018.* Austin: The University of Texas at Austin.

Barber, B., McLain, T., & Edwards, B. (2007). Vision-Based Landing of Fixed-WIng Miniature Air Vehicles. *Infotech@Aerospace.* Rohnert Park: AIAA.

Barrile, V., Candela, G., Fotia, A., & Bernardo, E. (2019). UAV Survey of Bridges and Viaduct: Workflow and Application. *ICCSA 2019: Computational Science and Its Applications*, (pp. 269-284).

Bharati, S. P., Wu, Y., Sui, Y., Padgett, C., & Wang, G. (2018). Real-Time Obstacle Detection and Tracking for Sense-and-Avoid Mechanism in UAVs. *IEEE Transactions on Intelligent Vehicles, 3*(2), 185-197.

Brownlee, J. (2019, September 16). *A Gentle Introduction to Transfer Learning for Deep Learning*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Burke, C., Rashman, M., Wich, S., Symons, A., Theron, C., & Longmore, S. (2019). Optimizing Observing Strategies for Monitoring Animals using Drone-Mounted Thermal Infrared Cameras. *International Journal of Remote Sensing*, 439-467.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 679-698.

Chand, B. N., Mahalakshmi, P., & Naidu, V. P. (2017). Sense and Avoid Technology in Unmanned Aerial Vehicles: A Review. *2017 International Conference on*

Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT). Mysuru: IEEE.

Chichella, V., Kaminer, I., Dobrokhodov, V., & Hovakimyan, N. (2013). Cooperative Vision Based Tracking of Multiple UAVs. *AIAA Guidance, Navigation, and Control (GNC) Conference.* Boston.

Choi, H., Geeves, M., Alsalam, B., & Gonzalez, F. (2016). Open Source Computer-Vision Based Guidance System for UAVs On-Board Decision Making. *2016 IEEE Aerospace Conference.* Big Sky: IEEE.

Cichella, V., Kaminer, I., Dobrokhodov, V., & Hovakimyan, N. (2013). Cooperative Vision Based Tracking of Multiple UAVs. *AIAA Guidance, Navigation, and Control (GNC) Conference.* Boston: AIAA.

Davies, I. (2013). Red Knot - Calidris cannutus. *Macaulay Library.* Cornell Lab of Ornithology, Ithaca. Retrieved from https://ebird.org/species/redkno

DJI. (2019, June 25). *Mavic 2*. Retrieved from DJI: https://www.dji.com/ca/mavic-2

Dolph, C., Logan, M., Glaab, L., Vranas, T., McSwain, R., Johns, Z., & Severance, K. (2017). Sense and Avoid for Small Unmanned Aircraft Systems. *AIAA SciTech Forum.* Grapevine: AIAA.

Dronecode. (2019, July 25). *Pixhawk 1 Flight Controller*. Retrieved from PX4 Docs: https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk.html

Ellenberg, A., Branco, L., Krick, A., Bartoli, I., & Kontsos, A. (2015). Use of Unmanned Aerial Vehicle for Quantitative Infrastructure Evaluation. *Journal of Infrastructure Systems, 21*(3).

Friestsch, N., Meister, O., Schlaile, C., Seibold, J., & Trommer, G. (2008). Vision Based Hovering and Landing System for a VTOL-MAV with Geo-Localization Capabilities. *AIAA Guidance, Navigation and Control Conference and Exhibit.* Honolulu: AIAA.

Frietsch, N., Seibold, J. C., Wei, M., & Trommer, G. (2011). Real Time Implementation of a Vision-Based UAV Detection and Tracking System for UAV-Navigation Aiding. *AIAA Guidance, Navigation, and Control Conference.* Portland: AIAA.

Gonzalez, L., Montes, G., Puig, E., Johnson, S., Mengersen, K., & Gaston, K. (2016). Unmanned Aerial Vehicles (UAVs) and Artificial Intelligence Revolutionizing Wildlife Monitoring and Conservation. *Sensors*.

Government of Canada. (2019, June 20). *Canadian Aviation Regulations (SOR/96-433) Part IX - Remotely Piloted Aircraft Systems.* Retrieved August 20, 2019, from https://laws-lois.justice.gc.ca/eng/regulations/SOR-96-433/page-164.html#h-1000299

Government Publishing Office. (2016, June 28). *Electronic Code of Federal Regulations - 14 CFR Part 107.* Retrieved August 20, 2019, from https://www.ecfr.gov/cgi-bin/text-idx?SID=5422bc0557b704edb9d20783d6f44847&mc=true&tpl=/ecfrbrowse/Title14/14cfr107_main_02.tpl

Gundlach, J. (2012). Designing Unmanned Aircraft Systems: A Comprehensive Approach. Reston: American Institute of Aeronautics and Astronautics, Inc.

Harris, C., & Stephens, M. (1988). A Combined Corner and Edge Detector. *Alvey Vision Conference.*

Hodgson, J., Baylis, S., Mott, R., Herrod, A., & Clarke, R. (2016). *Precision Wildlife Monitoring Using Unmanned Aerial Vehicles.* Scientific Reports.

Hosen, J., Helgesen, H., Fusini, L., Fossen, T., & Johansen, T. (2016). Vision-Aided Nonlinear Observer for Fixed-Wing Unmanned Aerial Vehicle Navigation. *Journal of Guidance, Control, and Dynamics*, 1777-1789.

Ichikawa, A., Abe, Y., Ikeda, T., Ohara, K., Kishikawa, J., Ashizawa, S., . . . Fukuda, T. (2017). UAV with Manipulator for Bridge Inspection - Hammering system for mounting to UAV -. *IEEE/SICE International Symposium on System Integration.* Taipei.

JARUS. (2019). *Publications*. Retrieved from Joint Authorities for Rulemaking on Unmanned Systems: http://jarus-rpas.org/publications

Johnson, J. (1958). Analysis of Image Forming Systems. *Image Intensifier Symposium*, (pp. 244-273). Ft. Belvoir, VA.

Jung, H.-J., Lee, J.-H., & Kim, I.-H. (2018). Challenging issues and solutions of bridge inspection technology using unmanned aerial vehicles. *SPIE Smart Structures and Materials + Nondestructive Evaluation and Health Monitoring.* Denver.

Kellenberger, B., Marcos, D., & Tuia, D. (2018). Detecting Mammals in UAV Images: Best Practices to Address a Substantially Imbalanced Dataset with Deep Learning. *Remote Sensing of Environment*, 139-153.

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2019, October 3). A Survey of Recent Architectures of Deep Convolutional Neural Networks. Nilore, Islamabad, Pakistan.

Library of Congress. (2019). *Regulation of Drones*. Retrieved from Library of Congress: https://www.loc.gov/law/help/regulation-of-drones/index.php

Lindeberg, T. (1998). Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*, 79-116.

Lipton, E. (2016). Least Sandpiper - Calidris minutilla. *Macaulay Library.* Cornell Lab of Ornithology, Ithaca. Retrieved from https://ebird.org/species/leasan

Luo, P., & Pei, H.-L. (2007). An Autonomous Helicopter with Vision Based Navigation. *IEEE International Conference on Control and Automation* (pp. 2595-2599). Guangzhou: IEEE.

Mallick, S. (2017, February 13). *Object Tracking using OpenCV (C++/Python)*. Retrieved from LearnOpenCV.com: https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

Martins, W., Ramos, A., Braga, R., Ribeiro, L., & Mora-Camino, F. (2017). Computer Vision in Remotely Piloted Aircraft (RPA) to Avoid Obstacles During Flight. *2017 18th International Conference on Advanced Robotics (ICAR)* (pp. 316-321). Hong Kong: IEEE.

Noble, F. K. (2016). Comparison of OpenCV's Feature Detectors and Feature Matchers. *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP).* Nanjing: IEEE.

Omar, T., & Nehdi, M. (2017). Remote sensing of concrete bridge decks using unmanned aerial vehicle infrared thermography. *Automatino in Construction, 83*, 360-371.

Opromolla, R., Vetrella, A. R., Fasano, G., & Accardo, D. (2018). Airborne Visual Tracking for Cooperative UAV Swarms. *AIAA SciTech Forum.* Kissimmeee: AIAA.

Ortiz, A. E., & Neogi, N. (2006). Color Optic Flow: A Computer Vision Approach for Object Detection on UAVs. *2006 IEEE/AIAA 25th Digital Avionics Systems Conference* (pp. 4C3-1 - 4C3-12). Portland: IEEE.

Ouaknine, A. (2018, February 5). *Review of Deep Learning Algorithms for Object Detection*. Retrieved from Zyl Story: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Owlia, S. (2013). *Real-Time Autonomous Obstacle Avoidance for Low-Altitude Fixed-Wing Aircraft.* Ottawa: Carleton University.

Peng, K., Shiyu, Z., Lin, F., & Chen, B. (2013). Vision based Target Tracking/Following and Estimation of Target Motion. *AIAA Guidance, Navigation, and Control (GNC) Conference.* Boston: AIAA.

Raspberry Pi Foundation. (2019, September 2). *Camera Module*. Retrieved from Raspberrypi.org: https://www.raspberrypi.org/documentation/hardware/camera/

Raspberry Pi Foundation. (2019, September 2). *Raspberry Pi 3 Model B+*. Retrieved from Raspberrypi.org: https://www.raspberrypi.org/documentation/hardware/camera/

Razzanelli, M., Innocenti, M., Pannocchia, G., & Pollini, L. (2019). Vision-based Model Predictive Control for Unmanned Aerial Vehicles Automatic Trajectory Generation and Tracking. *AIAA Scitech 2019 Forum.* San Diego: AIAA.

RCbenchmark. (2019, September 20). *Otus Tracker.* Retrieved from RCBenchmark.com: https://www.rcbenchmark.com/pages/otus-tracker

Rey, N., Volpi, M., Joost, S., & Tuia, D. (2017). Detecting Animals in African Savanna with UAVs and the Crowds. *Remote Sensing of Environment*, 341-351.

Rosebrock, A. (2018, July 30). *OpenCV Object Tracking*. Retrieved from PyImageSearch.com: https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/

Schulz, H.-W., Buschmann, M., Kruger, L., Winkler, S., & Vorsmann, P. (2005). Vision-Based Autonomous Landing for Small UAVs - First Experimental Results. *Infotech@Aerospace.* Arlington: AIAA.

SGL. (2019). *SGL Aircraft*. Retrieved from Sanders Geophysics Limited: http://www.sgl.com/Aircraft.html

Shi, J., & Tomasi, C. (1994). Good Features to Track. *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.* Seattle: IEEE.

Sridhar, B., & Cheng, V. (1988, June). Computer Vision Techniques for Rotorcraft Low-Altitude Flight. *IEEE Control Systems Magazine*, pp. 59-61.

Tuck, L. (2019). *Characterization and Compensation of Magnetic Interference Resulting from Unmanned Aircraft Systems.* Ottawa: Carleton University.

Ullman, S. (1976). *The Interpretation of Structure from Motion.* Boston: Massachusetts Institute of Technology.

Verbickas, R. (2012). *Convolutional Neural Network-based Vision Systems for Unmanned Aerial Vehicles.* Ottawa: Carleton University.

Vetrella, A. R., Fasano, G., Accardo, D., & Moccia, A. (2016). Differential GNSS and Vision-Based Tracking to Improve Navigation Performance in Cooperative Multi-UAV Systems. *Sensors*.

Yamada, M., Nakao, M., Hada, Y., & Sawasaki, N. (2017). Development and Field Test of Novel Two-Wheeled UAV for Bridge Inspections. *International Conference on Unmanned Aircraf Systems (ICUAS).* Miami.

Zhang, C., Chen, J., Song, C., & Xu, J. (2014). An UAV Navigation Aided with Computer Vision. *2014 26th Chinese Control and Decision Conference (CCDC)* (pp. 5297-5301). Changsha: IEEE.

Appendices

Appendix A - Random Image Set Sample from Kaggle Dataset

A.1 Cat Dataset

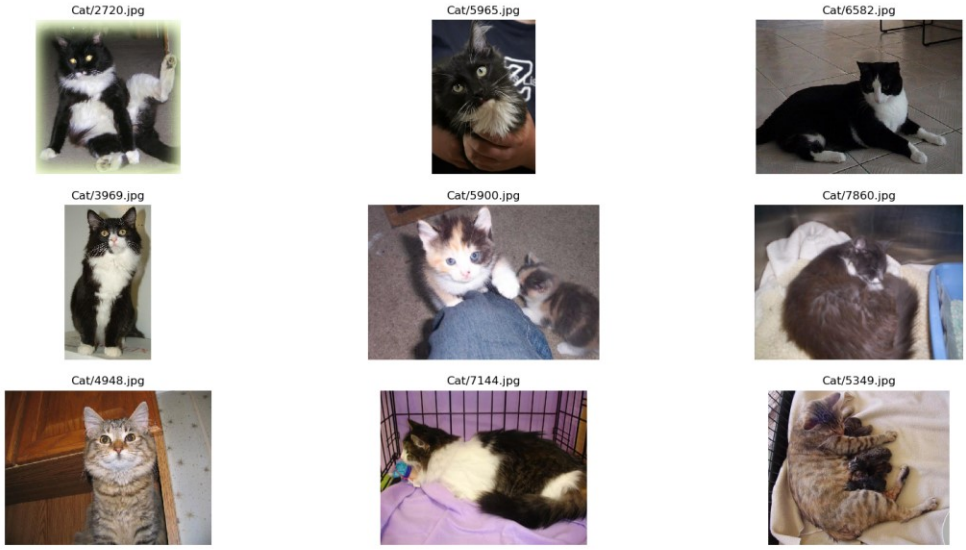

Figure A.1: Randomized image selection from cat training dataset

A.2 Dog Dataset

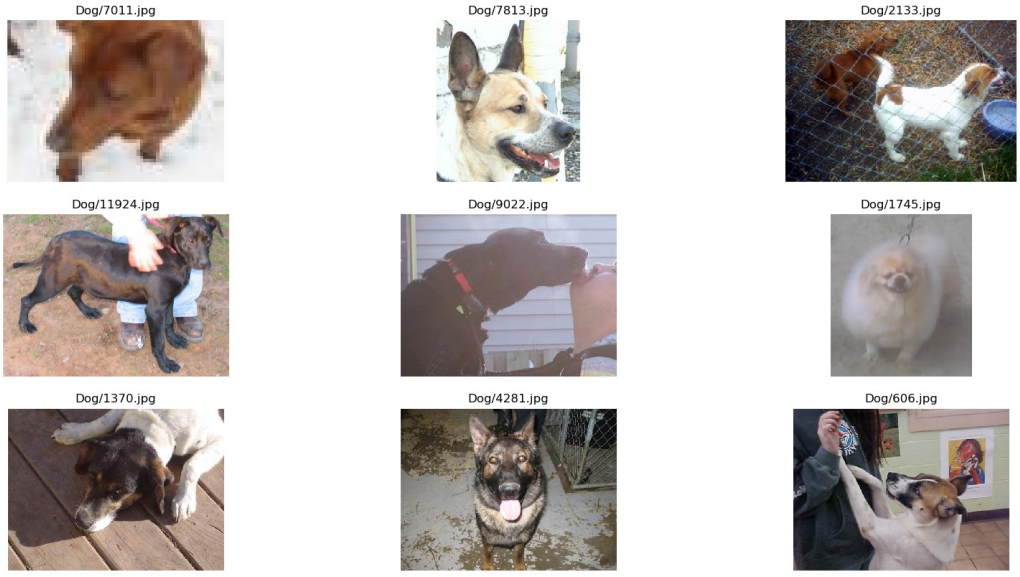

Figure A.2: Randomized image selection from dog training dataset

Appendix B - Sample of Data Collected from Vision-based Positioning System

Table B.1: Sample data measurements for Calibration Set 1

Test Point	Estimated (mm)	Laser Range Finder (mm)	VS Reading (mm)
1	200	207	223
			224
			223
			223
			222
2	400	428	442
			441
			442
			442
			442
3	600	610	626
			625
			626
			625
			625
4	800	813	829
			829
			829
			830
			831
5	1000	1009	1106
			1078
			1043
			1057
			1101
6	1200	1214	1512
			1564
			1667
			1531
			1599
7	1400	1404	1795
			1713
			1824
			1771
			1766

Table B.2: Position data collected by VPS for X-axis test - trial 3

FPS	Time (s)	x_pos	x_measured (cm)	y_pos	y_measured (cm)	z_pos	z_measured (cm)
	0.00	22.69	0.00	0.03	0.00	-6.55	0.00
5.93	0.17	22.69	0.01	0.03	0.00	-6.55	0.00
20.05	0.22	22.69	0.00	0.05	0.04	-6.55	0.00
26.39	0.26	22.71	0.04	0.13	0.23	-6.56	-0.02
29.92	0.29	22.79	0.22	0.28	0.56	-6.59	-0.10
10.88	0.38	23.44	1.65	0.88	1.87	-6.82	-0.61
8.16	0.50	23.43	1.64	0.88	1.87	-6.82	-0.61
23.32	0.55	23.44	1.65	0.88	1.87	-6.82	-0.61
7.01	0.69	23.44	1.64	0.88	1.87	-6.82	-0.61
8.87	0.80	23.44	1.65	0.88	1.87	-6.82	-0.61
11.39	0.89	23.44	1.64	0.88	1.87	-6.82	-0.61
10.13	0.99	23.44	1.64	0.88	1.87	-6.82	-0.61
7.37	1.12	23.44	1.65	0.88	1.87	-6.82	-0.61
6.47	1.28	23.44	1.64	0.88	1.87	-6.82	-0.61
7.77	1.41	23.44	1.65	0.88	1.87	-6.82	-0.61
11.14	1.50	23.44	1.65	0.88	1.87	-6.82	-0.61
11.39	1.59	23.44	1.65	0.88	1.87	-6.82	-0.61
8.22	1.71	23.44	1.65	0.88	1.87	-6.82	-0.61
8.94	1.82	23.44	1.64	0.88	1.87	-6.82	-0.61
12.35	1.90	23.44	1.64	0.88	1.87	-6.82	-0.61