2019-09

VISION-BASED RELATIVE POSITION ESTIMATION AND INTERCEPT TRAJECTORY PLANNING FOR SMALL UNMANNED AIRCRAFT SYSTEMS

Tan, Kang Hao

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

VISION-BASED RELATIVE POSITION ESTIMATION AND INTERCEPT TRAJECTORY PLANNING FOR SMALL UNMANNED AIRCRAFT SYSTEMS

by

Kang Hao Tan

September 2019

Thesis Advisor:	Oleg A. Yakimenko
Second Reader:	Fotis A. Papoulias

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
VISION-BASED RELATIVE POSITION ESTIMATION AND INTERCEPT TRAJECTORY PLANNING FOR SMALL UNMANNED AIRCRAFT SYSTEMS

5. FUNDING NUMBERS

6. AUTHOR(S) Kang Hao Tan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

The proliferation of unmanned aircraft systems (UAS) has contributed to the asymmetric threat of malevolent actors exploiting this technology for mischief or harm. Existing ground-based solutions are limited by line of sight, while human-operated responder drones can be less responsive and more labor-intensive. Hence, there is a capability requirement for autonomous vision-based pursuit and interception of unauthorized drones. To address this, the author developed a computer vision (CV) algorithm to detect, track and estimate the relative position and range of a hovering and moving airborne small UAS target in field conditions. CV-based measurements were compared against GPS data, to assess the range and angular estimation performance of the CV algorithm. Then, the CV-estimated range and angular information was processed by a flight control algorithm utilizing simple angular guidance principle to pursue and intercept the target. Field tests of the algorithm were done using a prototype drone. This research will inform the conceptual design and choice of hardware implementation for a commercial-off-the-shelf-based counter-UAS capability. More broadly, the research contributes to the body of knowledge in autonomous object tracking applications.

14. SUBJECT TERMS computer vision, object detection, object tracking, position estimation, navigation, trajectory, UAS, drone	**15. NUMBER OF PAGES** 137
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

VISION-BASED RELATIVE POSITION ESTIMATION AND INTERCEPT TRAJECTORY PLANNING FOR SMALL UNMANNED AIRCRAFT SYSTEMS

Kang Hao Tan
Major, Republic of Singapore Air Force
Bachelor of Engineering (Electrical Engineering),
National University of Singapore, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2019

Approved by: Oleg A. Yakimenko
Advisor

Fotis A. Papoulias
Second Reader

Ronald E. Giachetti
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The proliferation of unmanned aircraft systems (UAS) has contributed to the asymmetric threat of malevolent actors exploiting this technology for mischief or harm. Existing ground-based solutions are limited by line of sight, while human-operated responder drones can be less responsive and more labor-intensive. Hence, there is a capability requirement for autonomous vision-based pursuit and interception of unauthorized drones. To address this, the author developed a computer vision (CV) algorithm to detect, track and estimate the relative position and range of a hovering and moving airborne small UAS target in field conditions. CV-based measurements were compared against GPS data, to assess the range and angular estimation performance of the CV algorithm. Then, the CV-estimated range and angular information was processed by a flight control algorithm utilizing simple angular guidance principle to pursue and intercept the target. Field tests of the algorithm were done using a prototype drone. This research will inform the conceptual design and choice of hardware implementation for a commercial-off-the-shelf-based counter-UAS capability. More broadly, the research contributes to the body of knowledge in autonomous object tracking applications.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

BB	Bounding Box
COTS	Commercial Off-The-Shelf
CV	Computer Vision
EFFBD	Enhanced Functional Flow Block Diagram
EO	Electro-optical
FMU	Flight Management Unit
FOV	Field of View
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSD	Ground Sample Distance
HSV	Hue, Saturation, Value
MOUT	Military Operations on Urban Terrain
MSER	Maximally Stable Extremal Regions
OV	Operational View
RTK	Real-Time Kinematics
SURF	Speeded Up Robust Features
SV	Systems View
SWaP	Size, Weight and Power
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
VNC	Virtual Network Computing
YOLO	You Only Look Once

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The proliferation of unmanned aircraft system (UAS) technology has increased the ease with which malevolent actors can exploit relatively low-cost technology to cause harm or mischief. For example, the sighting of illegal drones near Heathrow and Gatwick airports in the United Kingdom severely disrupted air traffic as flights were grounded for safety considerations, causing thousands of passengers to be stranded at airports (BBC 2019). While counter-UAS systems already exist in the market, these systems also have inherent limitations. The pursuit of counter-UAS capability is a Department of Defense priority, and this has been reflected in increased funding for counter-UAS research and development. The budget for counter-UAS technology grew by 99 percent over the preceding year to exceed $1B in the FY 2019 Defense Budget request (Gettinger 2018). A counter-drone intrusion system utilizing small UAS to execute autonomous pursuit of the unauthorized drones can improve responsiveness during intrusion incidents and provide better line-of-sight advantages.

This thesis examines the use of commercial-off-the-shelf (COTS) 4K cameras to implement monocular visual estimation of the relative range and angular offset in azimuth and elevation of a nominal drone target from the optical axis of the camera. A computer vision (CV) algorithm is developed to detect and localize the position of the target within the camera's field of view (FOV). A simple flight guidance algorithm is subsequently developed to use information from the CV algorithm to generate motion command signals to allow the observing drone platform to autonomously pursue and intercept the target. Implementing these functionalities would form the foundation for a practical counter-UAS capability.

Past studies on object-following applications involving UAS did not directly address the counter-UAS scenario due to the conditions, types of targets and applications under consideration. This thesis aims to contribute further in the field of object-following applications for UAS by implementing computer vision and flight guidance algorithms to sense and pursue an airborne moving target in a field environment.

The system architecture for a counter-UAS capability is considered from the operational, functional, and physical perspectives. The functional architecture helps to structure the workflow of the algorithms while the physical architecture enumerates the key physical components needed to implement the system.

Given that the counter-UAS aerial vehicle has limited size, weight, and power (SWaP) and needs to pursue the target in real time, a CV algorithm with lower computational demand was considered for implementation in this thesis. This thesis uses a color-space segmentation approach to detect and localize the target position within the image frames of the video stream from the camera. The baseline scenario considers only the detection of a single airborne target against a sky background of largely homogeneous color. These simplifying assumptions allow the thesis to focus on the range and angular performance of a COTS-based camera.

The camera was calibrated with a cardboard target under indoor conditions to determine the empirical relationship between the pixel width of the target and the distance between the target and the camera. This relationship would be used to estimate the range of the drone target in subsequent field tests. The experimental setup for evaluating the effectiveness of monocular estimation of range and angular offset is centered on using a hovering unmanned aerial vehicle (UAV) with a camera payload to record the positions of another target UAV that has been programmed with pre-determined waypoints. The rationale for this setup is to realistically replicate the air-to-air encounter between the counter-UAS system and an intruder drone, and to use the recorded videos from such encounters to develop and evaluate the CV algorithm. Real-time kinematic (RTK) Global Positioning System (GPS) receivers were mounted on both drones to obtain accurate measurements of the relative positions between the drones. Field tests were conducted at Impossible City and Camp Roberts test ranges near Marina, CA, and Paso Robles, CA, respectively, but only meaningful GPS data was available for tests at Impossible City.

The range estimation performance was assessed against GPS-based distances at Impossible City and waypoint-based distances at Camp Roberts. The upper range bound of the proposed algorithm was established to be 0.95^{o} in terms of the angular size of the target, which for the small-size UAS with width of 33cm corresponds to about 20 m range

from the camera. For larger targets, such as the ScanEagle UAS featuring a 3.1 m wingspan (Huber 2018), the detection range would be in the order of approximately 190 m. Because of the way the bounding box is drawn, the box width is always larger than the target width. Consequently, the inverse relationship between the target width and target distance from camera means that the target distance is always underestimated. Hence, a correction factor needs to be introduced. The limited tests conducted within this study suggest that the correction factor can be as large as 0.55.

A symmetrical set of waypoints was planned to evaluate the angular estimation performance of the CV algorithm. At Impossible City, significant asymmetry in the actual waypoints flown by the target was visually observed in the video frames and confirmed by GPS measurements. This affected the evaluation of the angular estimation performance at Impossible City, as the asymmetry in spatial positioning of the target is confounded with true angular estimation.

The field tests were repeated at Camp Roberts with an additional set of waypoints flown at closer range to the camera. Evaluation of the angular estimation performance was compared solely against planned waypoints. For the nine waypoints observed at 20 m away from the camera, the azimuth errors range from 1.5^{o}–18.4^{o} in magnitude, while the elevation errors range from 0.1^{o}–5.6^{o} in magnitude. For waypoints observed at 10m away from the camera, the azimuth errors range from 3.9^{o}–11.4^{o} in magnitude, while the elevation errors range from 0.3^{o}–4.9^{o} in magnitude. The aforementioned results of the angular estimation did not reveal any dependency in the estimation error from the actual target position within the image frame, contrary to initial expectations. The results of the range estimation, however, showed that a correction factor is needed, depending on the prior camera calibration. This paves the way for future work to further investigate feasible techniques for accurate monocular range estimation. Alternatively, the target should always be positioned in the middle of the image frame where it is aligned with the optical axis via active gimbal control to reduce estimation errors.

It should be noted that the current implementation of the CV algorithm is not adaptive to different landscapes and sky conditions. Hence, imperfections in image processing resulted in residual artifacts in the image frame, which affected the accuracy of

the bounding box in tracking the true target position. Flight videos from Impossible City suffered from image artifacts and had to be post-processed to remove outliers, which would otherwise worsen the angular estimation accuracy. Such post-processing would not be feasible for real-time flight guidance. Other means of feature tracking were explored and the Maximally Stable Extremal Regions (MSER) feature detection algorithm was found to be promising. MSER feature tracking, however, lacks the visual cue for range estimation, unlike bounding boxes with finite area, height and width. A possible strategy is to employ both bounding box construction and MSER feature detection functions to improve tracking accuracy.

A prototype system was built to implement the counter-UAS concept and it successfully demonstrated vertical take-off/landing, rotational (yaw), and translational motion in field tests at Camp Roberts. Nonetheless, the yaw motion of the drone vehicle was not stable. This might be due to electromagnetic interference from sources on the airframe affecting the GPS self-localization performance. The implementation of the prototype system encountered several integration challenges with different degrees of mitigation.

Field testing of the flight guidance algorithm was conducted at Impossible City. Erratic motion of the drone was observed. This is attributed to the intermittent appearance of image artifacts in the video frame, which generated unwanted motion vector commands, causing the drone to navigate in an erratic manner. The flight tests also showed that the criteria for meeting the control objective needed to be more robust. One possible way to make the criteria more robust in response to artifacts is to factor in additional stopping conditions.

References

BBC. 2019. “Heathrow Airport Drone Investigated by Police and Military.” January 9, 2019. https://www.bbc.com/news/uk-46804425.

Gettinger, Dan. 2018. “Summary of Drone Spending in the FY 2019 Defense Budget Request.” Center for the Study of the Drone at Bard College. April 9, 2018. https://dronecenter.bard.edu/files/2018/04/CSD-Drone-Spending-FY19-Web-1.pdf.

Huber, Mark. 2018. “ScanEagle UAS Offers New Capabilities.” AINonline. February 3, 2018. https://www.ainonline.com/aviation-news/defense/2018-02-03/scaneagle-uas-offers-new-capabilities.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my family members for their unwavering support and understanding over the past one year. I would like to express my deep gratitude to my thesis advisor, Professor Oleg Yakimenko, who provided a lot of support and guidance for my thesis and cultivated my interest in the MATLAB software and computer vision. My learning curve during the thesis journey would have been far steeper if not for the help, resources, and coaching that Mr. Rushen Dal, the engineering technician in the SE Department, had given me. In between writing my thesis and conducting field work, I am glad to have had the company of my SE classmates CQ, J, and JM, who motivated me to demand higher standards for myself and to polish my work. Last, but not least, kudos to my housemates who kept me well-fed with home-cooked meals and gave me a sense of home away from home.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This chapter outlines the background and the impetus for the thesis research, followed by the problem statement that the thesis aims to address.

A. BACKGROUND

There has been an increase in the usage of drones across a variety of industries ranging from defense applications to commercial sectors, such as cargo delivery and crop monitoring in agriculture. The proliferation of unmanned aircraft system (UAS) technology has brought with it many benefits, such as increased automation to relieve manpower shortages as well as manned-unmanned teaming for greater productivity and efficiency in labor-intensive tasks such as search and rescue missions. Yet, the flip side of the low barrier to UAS acquisition has meant that UAS technology is also easily available to malevolent actors who can exploit low-cost commercial off-the-shelf (COTS) UAS for asymmetric advantages in causing harm or mischief. For example, from December 2018 to January 2019, the sighting of illegal drones flying in the vicinity of Heathrow and Gatwick airports in the United Kingdom caused massive disruption to air traffic as flights were grounded, causing thousands of passengers to be stranded at airports (BBC 2019).

Existing counter-UAS systems in the market range from ground-based effectors relying on line-of-sight to remotely controlled drones with effector payloads such as nets (Khoe 2018). There are, however, inherent drawbacks to such solutions. Ground-based effectors are constrained by line-of-sight, and they may be further handicapped by effector range limitations, as the intruding drone can simply fly at higher altitudes to avoid being targeted. Mobile ground-based effectors can face terrain constraints, such as the presence of obstacles, water bodies, or sensitive and restricted areas. The sheer size of the facility that needs to be protected can also render static deployments of such effectors unfeasible and not cost-effective. Remotely controlled drones suffer from the same drawbacks, as the human operators are also ground-based personnel similarly constrained by line-of-sight and terrain. Further financial limitations appear as the scale of the proposed fleet of remotely controlled drones increases, as operators will require compensation for their

labor. These costs further escalate if there are requirements for round-the-clock facility protection.

The use of autonomous UAS in facility protection for countering unauthorized drone intrusions is a possible solution as the response entity, an aerial vehicle, is airborne and does not suffer the same degree of limitations imposed by line-of-sight and terrain. Labor costs considerations can be mitigated since the autonomous capability would minimize the size of the workforce needed to control the whole fleet in real time.

Compared to remotely controlled variants, a counter-drone intrusion system utilizing small UAS to execute autonomous pursuit of unauthorized drones can improve responsiveness during intrusion incidents, and enjoy better line-of-sight advantages compared to ground-based effectors. In the context of facility protection for airports, a mobile relatively low-cost COTS-based autonomous UAS solution can mitigate the challenges of continuously protecting large volumes of airspace. Realizing the autonomous pursuit capability will require the UAS to detect and localize the suspect drone and make use of spatial awareness to plan and execute a trajectory to pursue and intercept the moving target.

B. STATE OF THE ART IN OBJECT DETECTION / LOCALIZATION

This section details the review of literature from previous studies conducted by scholars and researchers in the field that contributed to this thesis research.

1. UAS in Object-Following Applications

Past studies on object-following applications involving UAS have been done on ground targets (Liu et al. 2017, 1–12). The focus of some of the previous research was to studying precision landing) or simple, non-maneuvering airborne objects (for example, balloons) (Mondragón et al. 2011, 1–7). The conditions and applications of such studies are not directly applicable to the purpose of realizing a UAS-based autonomous pursuit capability. This thesis aims to contribute further in the field of object-following applications for UAS by implementing a computer vision algorithm to sense and pursue an airborne moving target in a field environment.

One of the critical functions that the counter-UAS systems needs to perform is the ability to sense the position of the target. The electro-optical (EO) sensor will generate a stream of image frames which are then processed by a computer vision (CV) algorithm to detect the presence of the target in the image frames, and to localize the target within each frame, typically with a bounding box (for ease of illustration to human observers). Furthermore, for a moving target, the desired CV algorithm must track the movement of the target across successive image frames. According to a study by Parekh, Thakore, and Jaliya (2014, 2971), tracking is defined "as the problem of approximating the path of an object in the image plane as it moves around a scene."

A 2013 *International Journal of Computer Applications* article describes a variety of techniques for object detection and localization in the field of computer vision (Shantaiya, Verma, and Mehta 2013, 14–20)—such techniques are classified under four broad approaches: (1) feature-based (for example, shape, color), (2) template-based, (3) classifier-based, and (4) motion-based. In feature-based approaches, the goal is to segment the object from the background by means of image processing techniques to extract distinct object features such as corners, edges, and color. Template-based approaches rely on the availability of a given template describing a specific object, and the authors of the same article pointed out that "object detection becomes a process of matching features between the template and the image sequence under analysis" (Shantaiya, Verma, and Mehta 2013, 16). Classifier-based object detection approaches have gained popularity in recent years as advances in computing power have made deep learning accessible to CV researchers. One key drawback of the classifier approach, however, is the need to build up a sufficiently large set (for example, in the order of 300–400 different images per category of object for the You-Only-Look-Once (YOLO) deep learning algorithm) of annotated training data with positive and negative sample images to train the classifier algorithm to recognize the object with a high degree of confidence (Teo 2018). Motion-based approaches such as background subtraction and optical flow typically compare successive frames with an initial background frame to detect the motion of objects. These motion-based approaches perform best with stationary backgrounds (hence implying a static sensor) and are not suited for the context of a moving UAS pursuing a maneuvering intruder drone.

The low computational cost (Shantaiya, Verma, and Mehta 2013, 14–20) and relative simplicity in implementing a featured-based approach for object detection and localization makes it a feasible option for implementation on a small COTS-based UAS platform with finite size, weight, and power (SWaP) constraints. The focus of this thesis is on using detected visual cues to generate a flight path to pursue and home in on an intruding drone, thereby demonstrating the implementation of a vision-based relative position estimation and pursuit capability for the counter-UAS system. Improvements in object detection and localization performance are left as areas for future research.

2. Range Estimation

With a visual sensor, the position of the detected target in three-dimensional space is projected onto the two-dimensional image plane of the sensor. While such 2-D images offer information on the relative latitude and longitude offset in the image plane between the target and the observer, spatial awareness is not complete, as the range or depth information is missing from such images. Typically, the solution to range/depth estimation problems in computer vision and robotics applications is to employ stereo vision (Saxena, Schulte, and Ng 2007, 2197), in which the depth of a target object is estimated by triangulation using images from two cameras mounted at a fixed and known distance apart. The same study, however, also points out that there are limitations to stereo vision: first, inaccuracies in depth estimates tend to occur when the distances considered are large; and second, stereo vision performs poorly "for textureless regions of images where correspondences cannot be reliably found" (Saxena, Schulte, and Ng 2007, 2197). Implementation of dual sensors to achieve stereo vision also imposes penalties on the payload, the power consumption, and the computation load, given that the UAS involves an airborne platform with a finite payload capacity and power.

The same authors, however, also pointed out that the problem of range/depth estimation can be simplified with monocular cues (Saxena, Schulte, and Ng 2007, 2197), for example, if the size of the target object is known prior. This simplifies the implementation into a single sensor system and avoids the penalties on the payload, the power consumption, and the computation load. The dimensions of the representative drone

threat in an unauthorized drone intrusions incident can be reasonably bounded within a narrow range. Here, several assumptions are made. The first is that the intruding drone would need to be above a certain size to have meaningful endurance in the air to carry out its mission. This would require a sufficient power supply that would necessarily affect lift. The second assumption is that larger and more capable "professional grade" drones like the types employed in commercial videography tend to be more expensive, putting them out of reach of the average consumer. The larger drones are also more conspicuous, and less likely to be employed as a stealthy intruding platform.

A survey of the drone dimensions from leading drone manufacturers such as DJI, Yuneec, and Parrot shows that these products range from 214 mm to 322 mm in length with flight times ranging from 25 to 33 minutes (DJI n.d.; Parrot n.d.; Yuneec n.d.). In summary, this study considers consumer-grade drones designed and marketed as portable videography tools as the nominal representative intruder threat. In this study, the 3DR Solo (with a length of 330 mm and flight time of 25 minutes) is used to test and evaluate the CV algorithm.

C. PROBLEM FORMULATION AND THESIS STRUCTURE

Based on the review of the state-of-the-art in object detection/localization, it becomes clear that the problem of countering UAS intrusion with UAS-based solutions has not yet been fully addressed. This thesis's objective is to contribute to the solution of this problem by developing the prototype algorithm to detect and estimate the relative position of an airborne object representative of the nominal threat, and conducting a series of field tests employing small COTS drones to verify the effectiveness of the developed algorithms. Another thesis objective is to try to close the control loop passing the processed video information to the autopilot to make corrective actions, specifically, to navigate towards the detected airspace intruder. To this end, the specific research questions to be addressed are:

- What is the range detection limit of the developed CV algorithm?
- What is the accuracy of the monocular range and angular estimation technique?

- What are the dependencies in the estimation errors?
- What is the feasibility of the flight guidance algorithm utilizing the processed video information?

The remainder of this thesis is organized as follows: Chapter II outlines the system architecture of the envisioned counter-UAS capability from the standpoint of facility protection. Chapter III details the development of the feature-based CV algorithm for object detection and localization, and Chapter IV describes the calibration process for visual estimation of the target range. Chapter V presents the experimental setup to obtain Global Positioning System (GPS)-based range measurements for comparison against visual estimates. Chapter VI discusses and analyzes the results of the range and angular evaluations. Chapter VII details the implementation of a prototype system, implementation challenges encountered, as well as field tests of the prototype system. Finally, in Chapter VIII, the conclusions and recommendations are made.

II. SYSTEM ARCHITECTURE

This chapter describes the system architecture of the counter-UAS capability from the operational, functional and physical perspectives.

A. SYSTEM DESCRIPTION

The system of interest is a counter-UAS system to detect, pursue and intercept an unauthorized UAS which intrudes by flying into a protected facility. The main hardware product of the counter-UAS system is a small quadrotor unmanned aerial vehicle (UAV)—built from COTS component—which flies and lands on its own without the need for a separate launch/recovery system (unlike bigger types of UAS). The vehicle is mounted with a COTS camera, which functions as the EO sensor payload to allow the counter-UAS system to detect and track the target. The UAV's ability to fly and operate over long distances offers the end user non-line-of-sight advantages over areas of interest and beyond visual range.

The UAV can be controlled manually by a human operator—akin to remote-control aircraft—or flown via autopilot through a desired flight path with pre-programmed mission waypoints and minimal human effort. The operator controls the vehicle manually with a wireless remote controller. The remote controller has toggle controls to switch the UAV into autonomous mode and back into manual mode when required. Manual or automatic flight by waypoints allows the end user to direct the UAV to an area of interest (for example, based on early warning from perimeter sensors or intelligence) before the camera on the UAV starts to detect any potential intruding drone. This design consideration also factors in the finite detection range of the camera. Mission planning, downloading of waypoints and troubleshooting of software (onboard the UAV) are done on a laptop that is connected to the onboard computer of the UAV. An operational view (OV) OV-1 diagram illustrating the counter-UAS capability is shown in Figure 1.

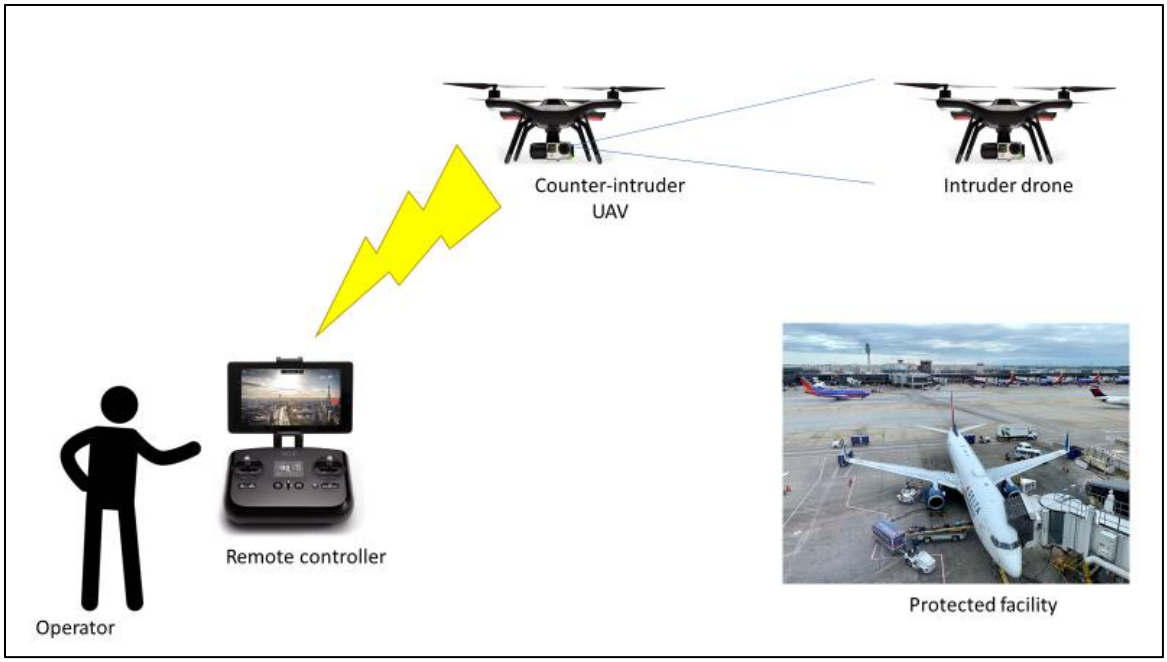


Figure 1. OV-1 Diagram for Counter-UAS Capability. Adapted from 3D Robotics (n.d.); 3D Robotics (2015); Murph (2019).

B. CAPABILITY VIEW

The counter-UAS system performs the mission of intercepting an intruding drone. The mission is achieved by the operational activities modeled in Figure 2.

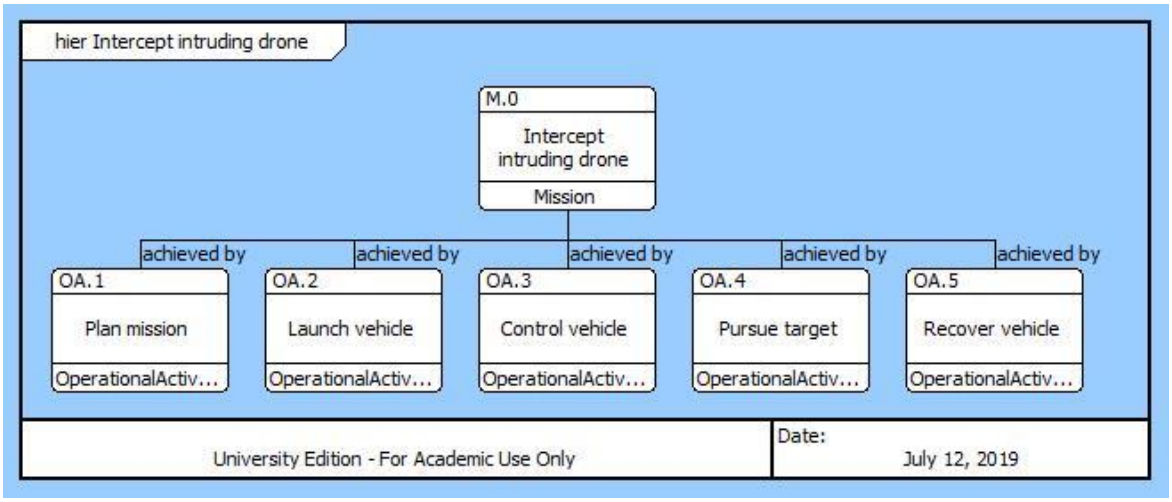


Figure 2. Mission and Operational Activities

The counter-UAS system provides the end user with the capability of intercepting an intruder drone in a protected area. This capability is refined into several lower echelon capabilities identified in Figure 3. These lower echelon capabilities are implemented by the system requirements shown in the same figure. This thesis focuses on the development of CV and guidance algorithms to implement requirements R.8 (provide autonomous operation), R.12 (detect airborne target), and R.13 (pursue target until successful interception).

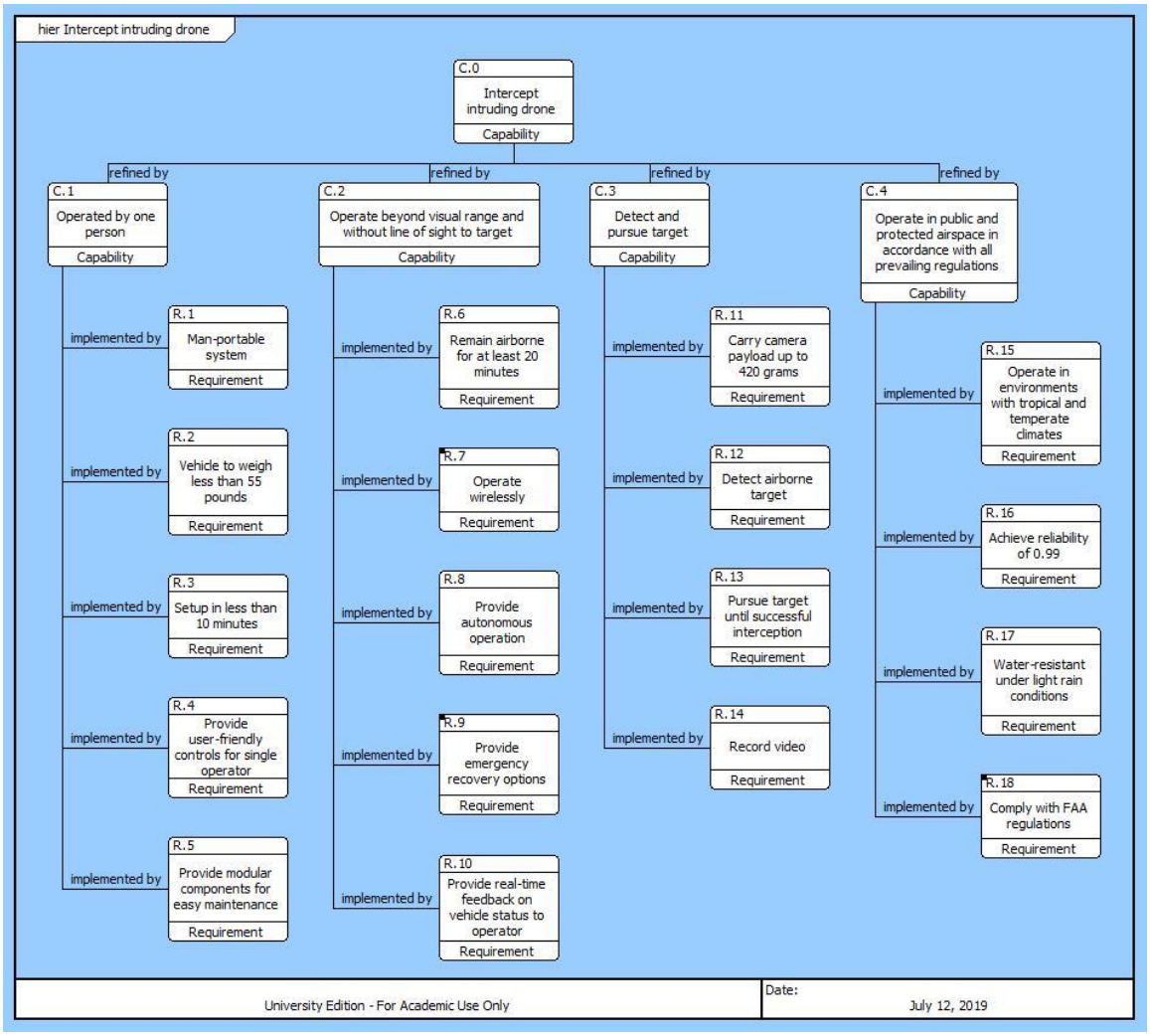


Figure 3. Capability Needs and System Requirements

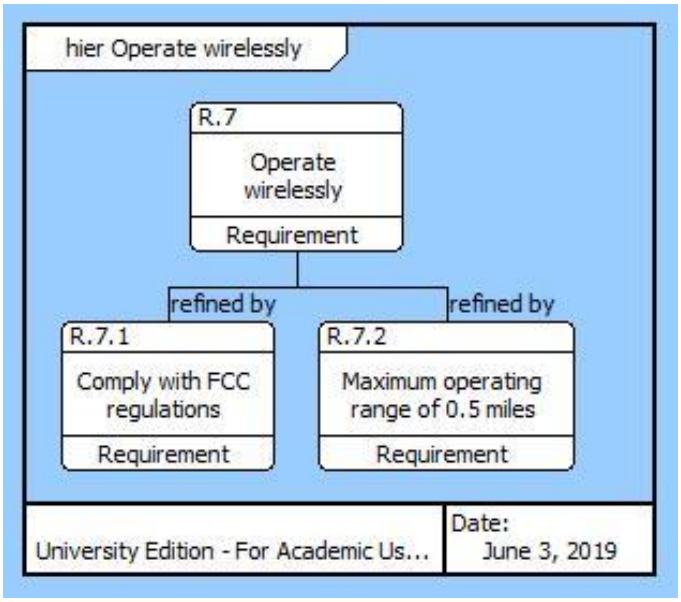


Figure 4. Expanded View of R.7

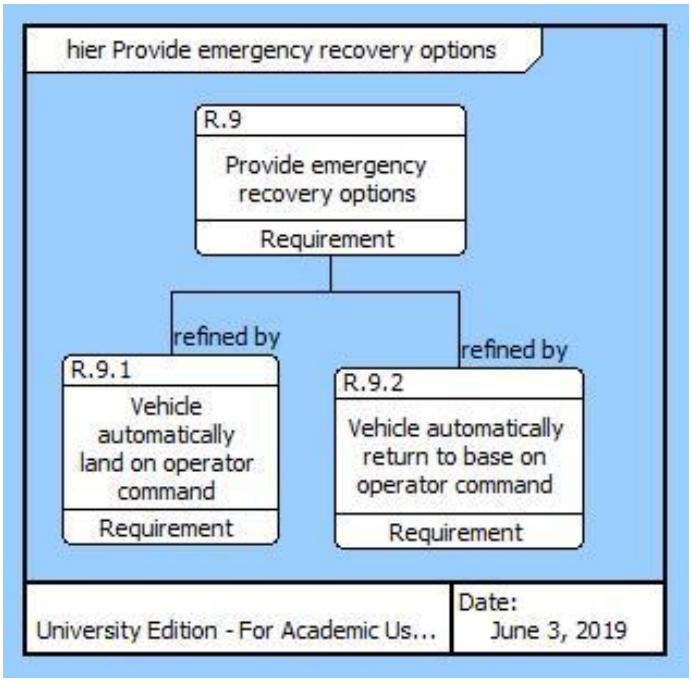


Figure 5. Expanded View of R.9

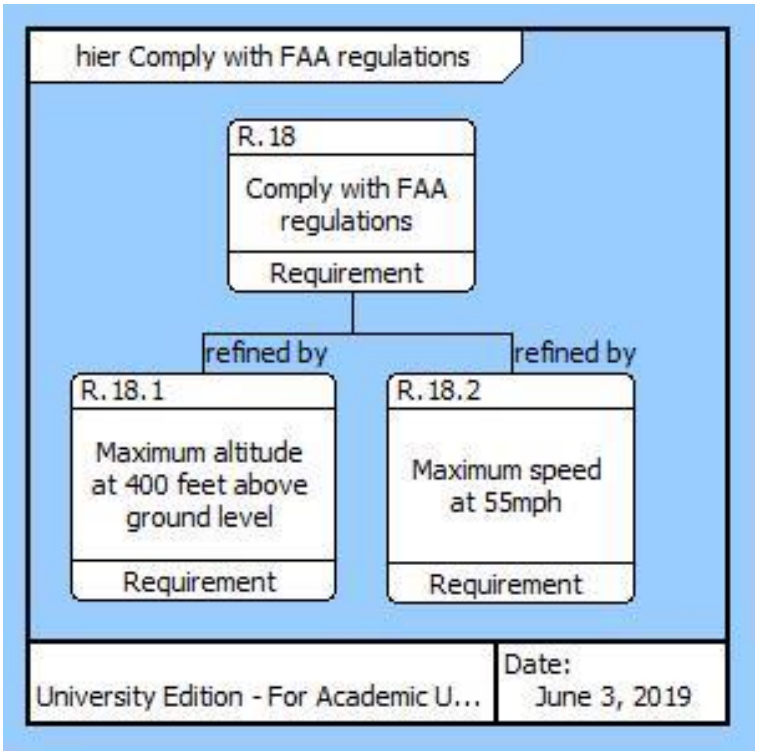


Figure 6. Expanded View of R.18

C. OPERATIONAL ARCHITECTURE

The operational flow of the system is illustrated in Figure 7. The diagram shows the resource flows into the system, through a series of operational activities, eventually culminating in the intended outcome—mission completion/termination. The input and output of items into/from the various operational activities are also outlined in the same figure.

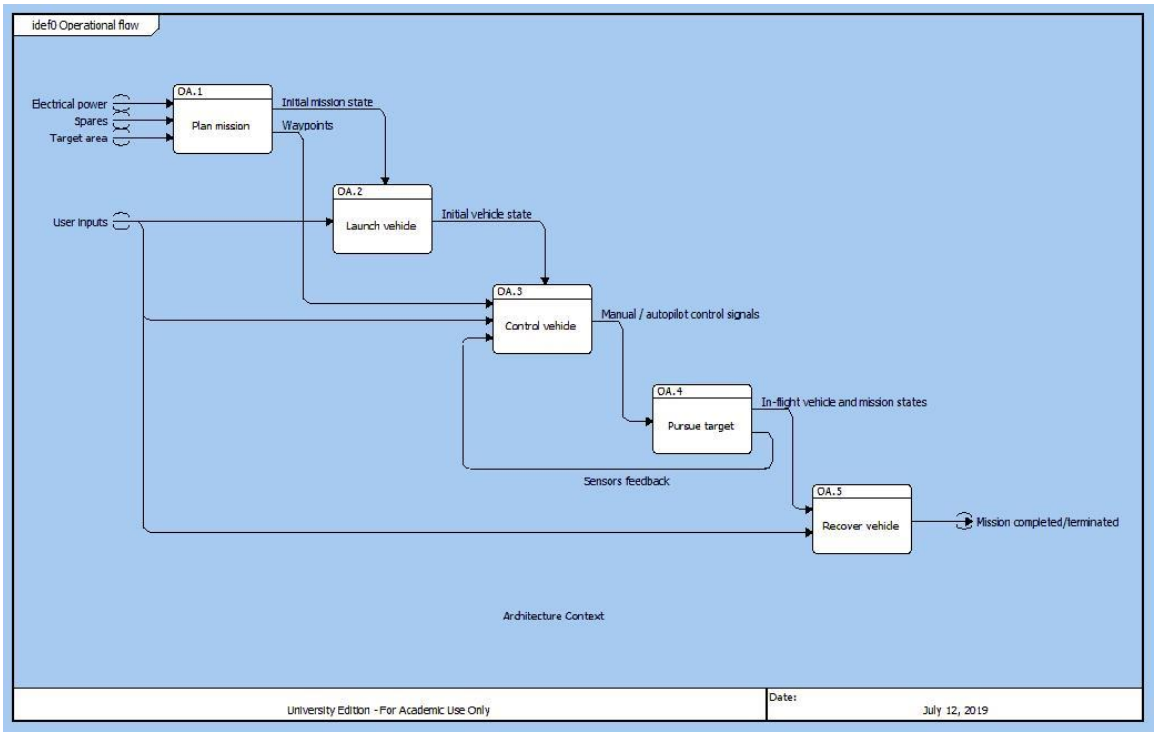


Figure 7. Operational Flow

A traceability matrix in Figure 8 shows how the various system functions implement the operational activities.

			Functions																						
			F.1.1	F.1.2	F.2.1	F.2.2	F.2.3	F.2.4	F.3.1	F.3.2	F.3.3	F.3.4	F.3.5	F.4.1	F.4.2	F.4.3	F.4.4	F.4.5	F.4.6	F.5.1	F.5.2	F.5.3	F.5.4	F.5.5
			Supply power	Store power	Accept mission inputs	Upload mission inputs to vehicle	Receive mission acknowledgment	Display mission acknowledgment	Accept manual control inputs	Send control inputs to vehicle	Receive control acknowledgment	Display control acknowledgment	Perform autopilot	Estimate current state	Display current state	Check vehicle and mission goal states	Generate lift	Generate propulsion	Provide directional control	Support camera on vehicle	Capture video	Detect target	Track target	Store video footage
Operational activity	OA.1	Plan mission	**1**	**1**	**1**	**1**	**1**	**1**																
	OA.2	Launch vehicle							**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**								
	OA.3	Control vehicle							**1**	**1**	**1**	**1**	**1**											
	OA.4	Pursue target											**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	OA.5	Recover vehicle							**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**								

Figure 8. Operational Activity to System Functions Traceability Matrix

D. FUNCTIONAL ARCHITECTURE

The functional hierarchy of the counter-UAS system is shown in Figure 9. The expanded views for functions F.3.1 and F.4.1 are shown in Figures 10 and 11, respectively.

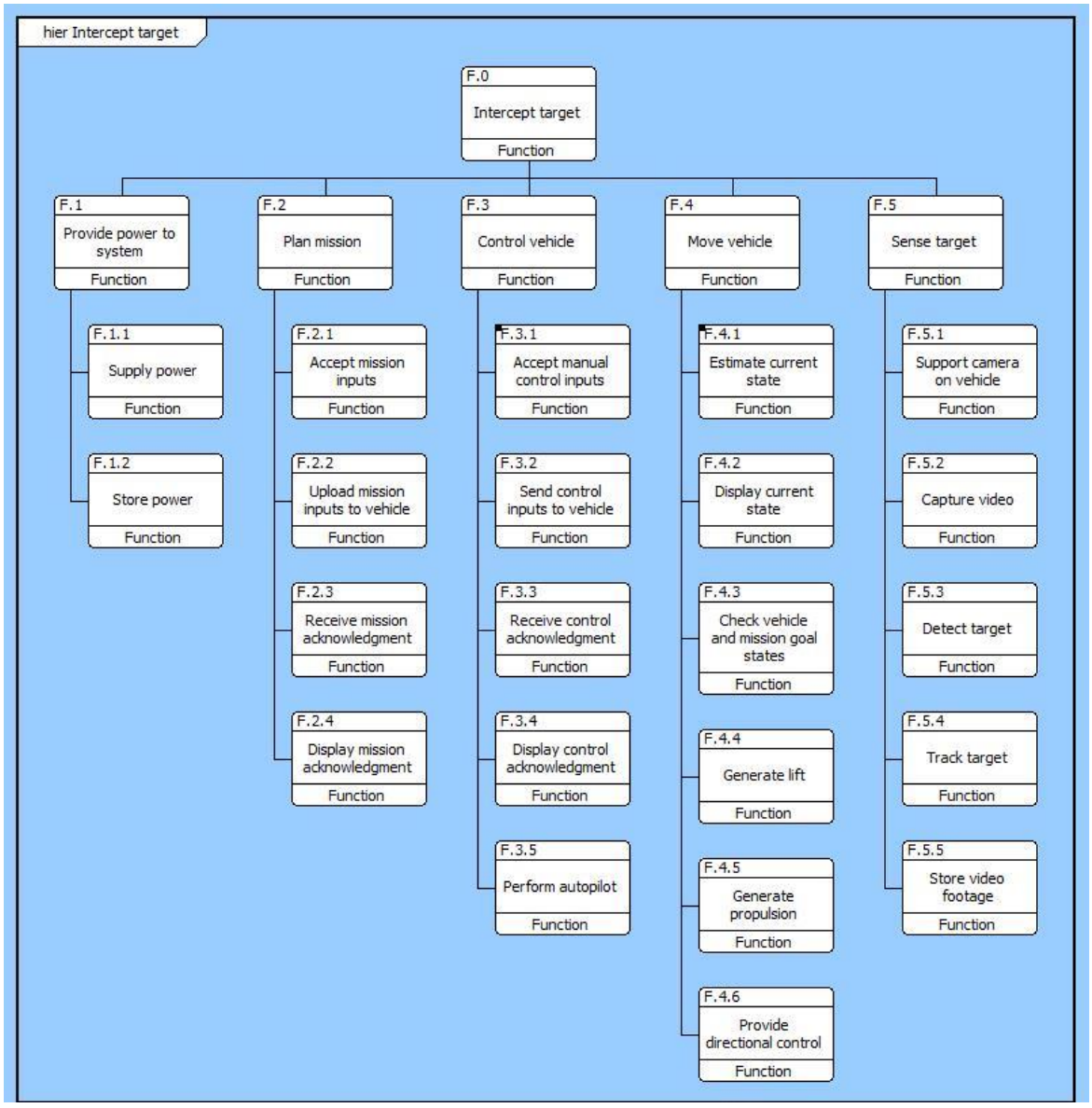


Figure 9. Functional Hierarchy

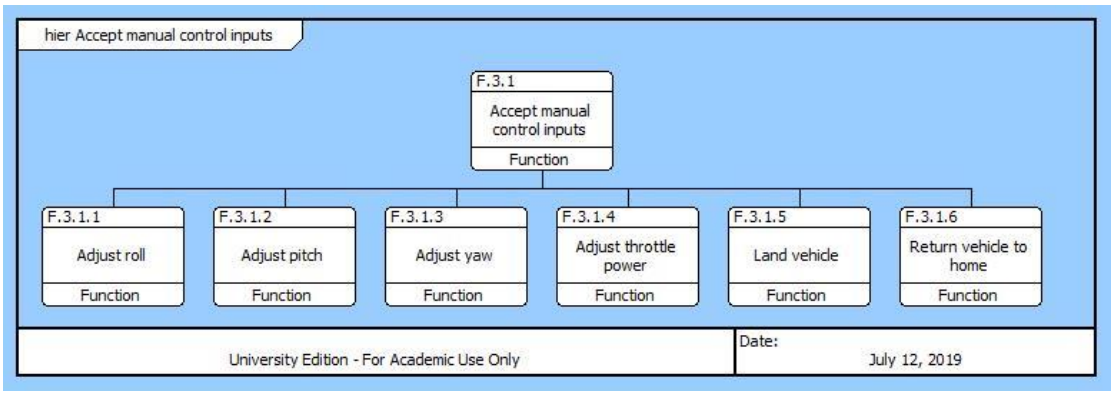


Figure 10. Expanded View for F.3.1

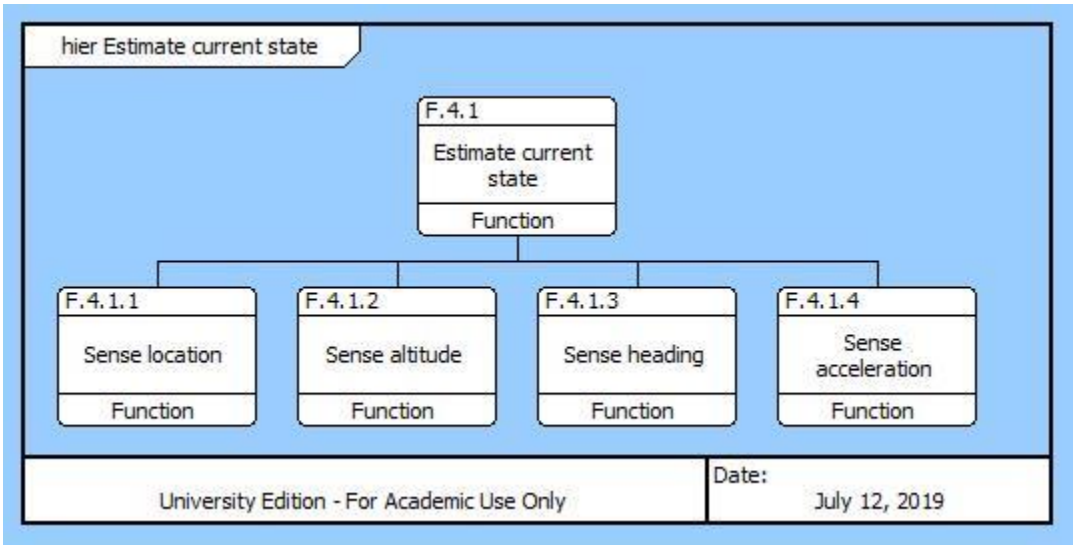


Figure 11. Expanded View for F.4.1

The functional behavior model of the system is further illustrated using Enhanced Functional Flow Block Diagram (EFFBD) diagrams. The top-level functions are shown in Figure 12. The loop enclosing functions F.3–F.5 represents the mission control loop. Once an initial mission is planned, the counter-UAS system iteratively awaits control inputs, moves the vehicle to an area of interest based on manual commands or pre-programmed waypoints, and starts to detect and pursue the target within the area of interest until the mission is completed or terminated.

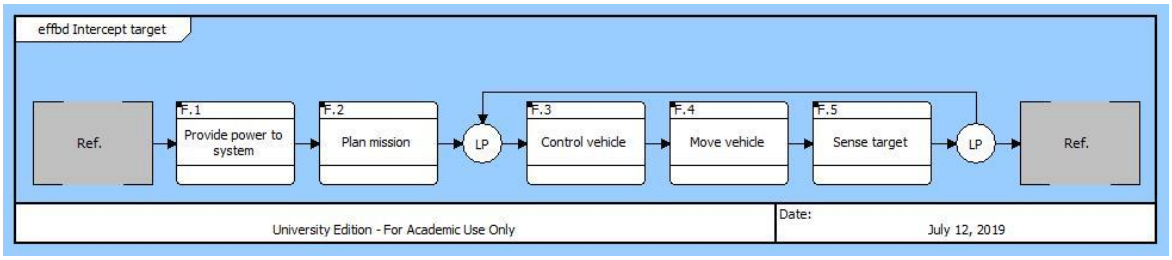


Figure 12. Top-level Functions

Functions F.1–F.5 are further decomposed into their respective lower echelon functions in Figures 13–17. The loop enclosing all the functions in Figure 12 represents the flight control loop, and it is responsible for steering and flying the vehicle along the waypoints to complete the flight path. A change in the mission state (for example, completion of the last waypoint) or vehicle state (for example, low battery level or user activated emergency landing) can cause the loop to stop iterating. The inputs and outputs between functions are shown in the green boxes of the following figures.

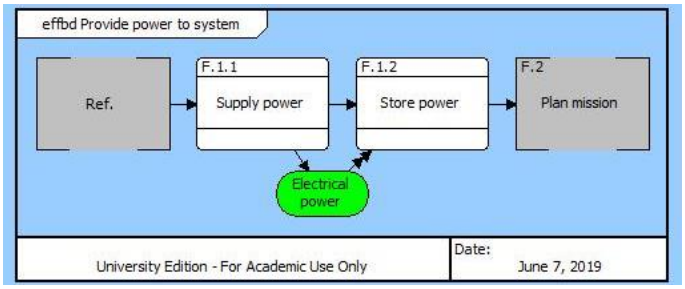


Figure 13. Expanded View of F.1

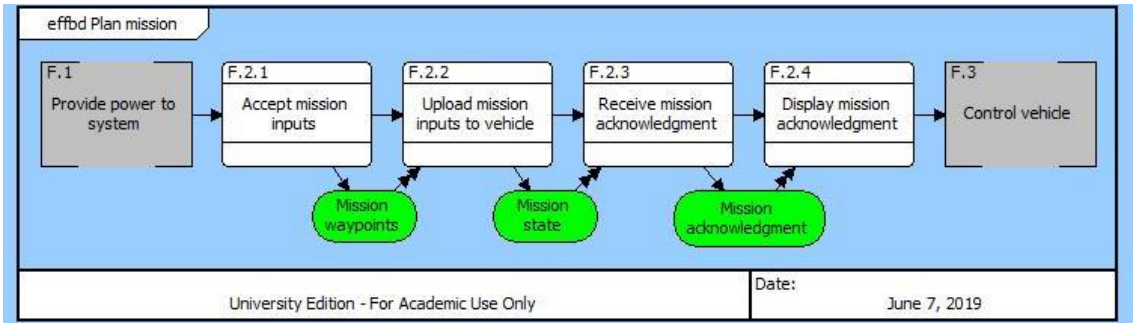


Figure 14. Expanded View of F.2

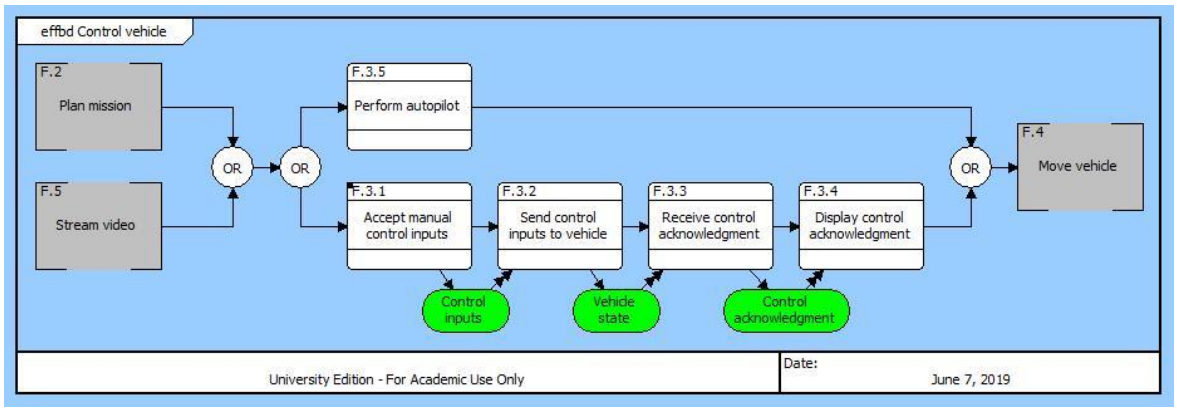


Figure 15. Expanded View of F.3

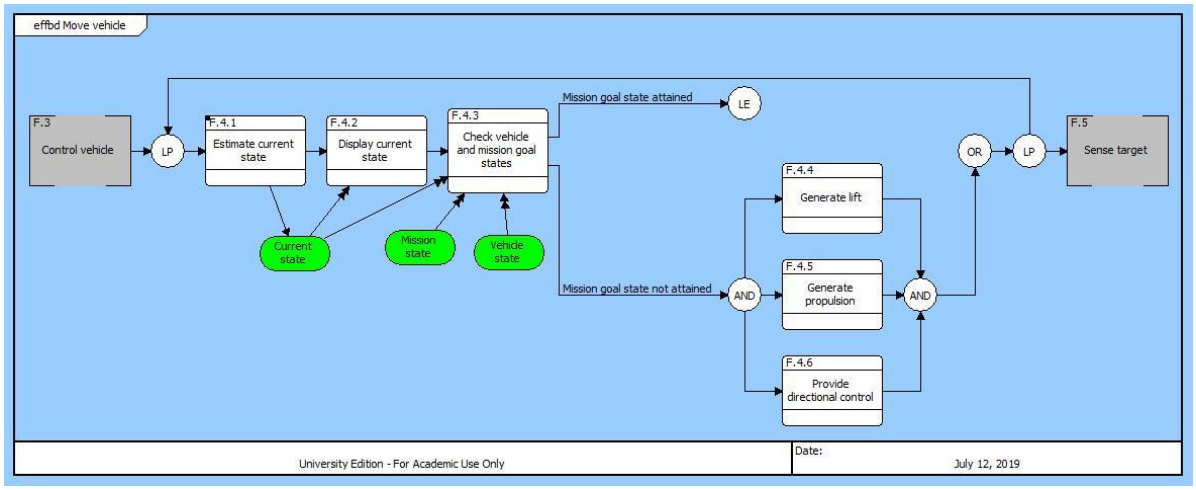


Figure 16. Expanded View of F.4

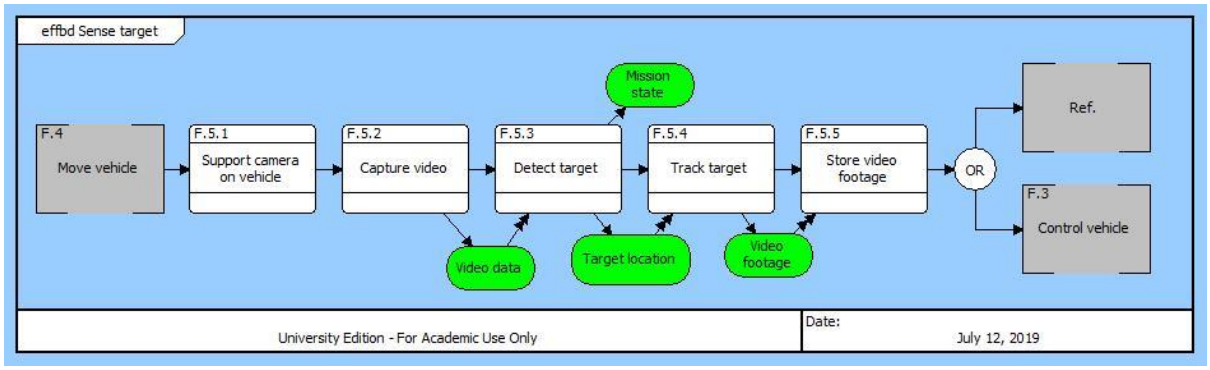


Figure 17. Expanded View of F.5

A traceability matrix in Figure 18 illustrates how the various system functions are allocated to physical components. Each function is allocated to at least one component.

			Functions																								
			F.1.1	F.1.2	F.2.1	F.2.2	F.2.3	F.2.4	F.3.1	F.3.2	F.3.3	F.3.4	F.3.5	F.4.1	F.4.2	F.4.3	F.4.4	F.4.5	F.4.6	F.5.1	F.5.2	F.5.3	F.5.4	F.5.5
			Supply power	Store power	Accept mission inputs	Upload mission inputs to vehicle	Receive mission acknowledgment	Display mission acknowledgment	Accept manual control inputs	Send control inputs to vehicle	Receive control acknowledgment	Display control acknowledgment	Perform autopilot	Estimate current state	Display current state	Check vehicle and mission goal states	Generate lift	Generate propulsion	Provide directional control	Support camera on vehicle	Capture video	Detect target	Track target	Store video footage
Physical components	SYS.1.1	Touchscreen display			1			1							1									
	SYS.1.2	Mission planning software application			1	1	1	1							1									
	SYS.1.3	USB port				1	1																	
	SYS.1.4	Device battery		1																				
	SYS.1.5	Device charging cable	1																					
	SYS.1.6	Device microprocessor			1	1	1	1							1									
	SYS.2.1	Dual-axis joysticks							1															
	SYS.2.2	Input buttons							1															
	SYS.2.3	RC display screen										1			1									
	SYS.2.4	Ground RC radio				1	1			1	1	1												
	SYS.2.5	RC battery		1																				
	SYS.2.6	RC charging cable	1																					
	SYS.2.7	RC microprocessor				1	1		1	1	1	1												
	SYS.3.1	Flight management unit (processor)				1	1			1	1		1	1		1								
	SYS.3.2	GPS module												1										
	SYS.3.3	Barometer												1										
	SYS.3.4	Compass												1										
	SYS.3.5	Gyroscope												1										
	SYS.3.6	Airframe RC radio				1	1			1	1													
	SYS.3.7	Rotors															1	1	1					
	SYS.3.8	Vehicle battery		1																				
	SYS.3.9	Vehicle battery charging cable	1																					
	SYS.3.10	FMU USB port				1	1																	
	SYS.3.11	On-board computer																				1	1	
	SYS.4.1	HDMI bridge module																		1	1			
	SYS.4.2	Camera																			1			1
	SYS.4.3	Camera battery		1																				
	SYS.4.4	Camera charging cable	1																					

Figure 18. Physical Components to System Functions Traceability Matrix

E. PHYSICAL ARCHITECTURE

The physical hierarchy of the counter-UAS system is shown in Figure 19.

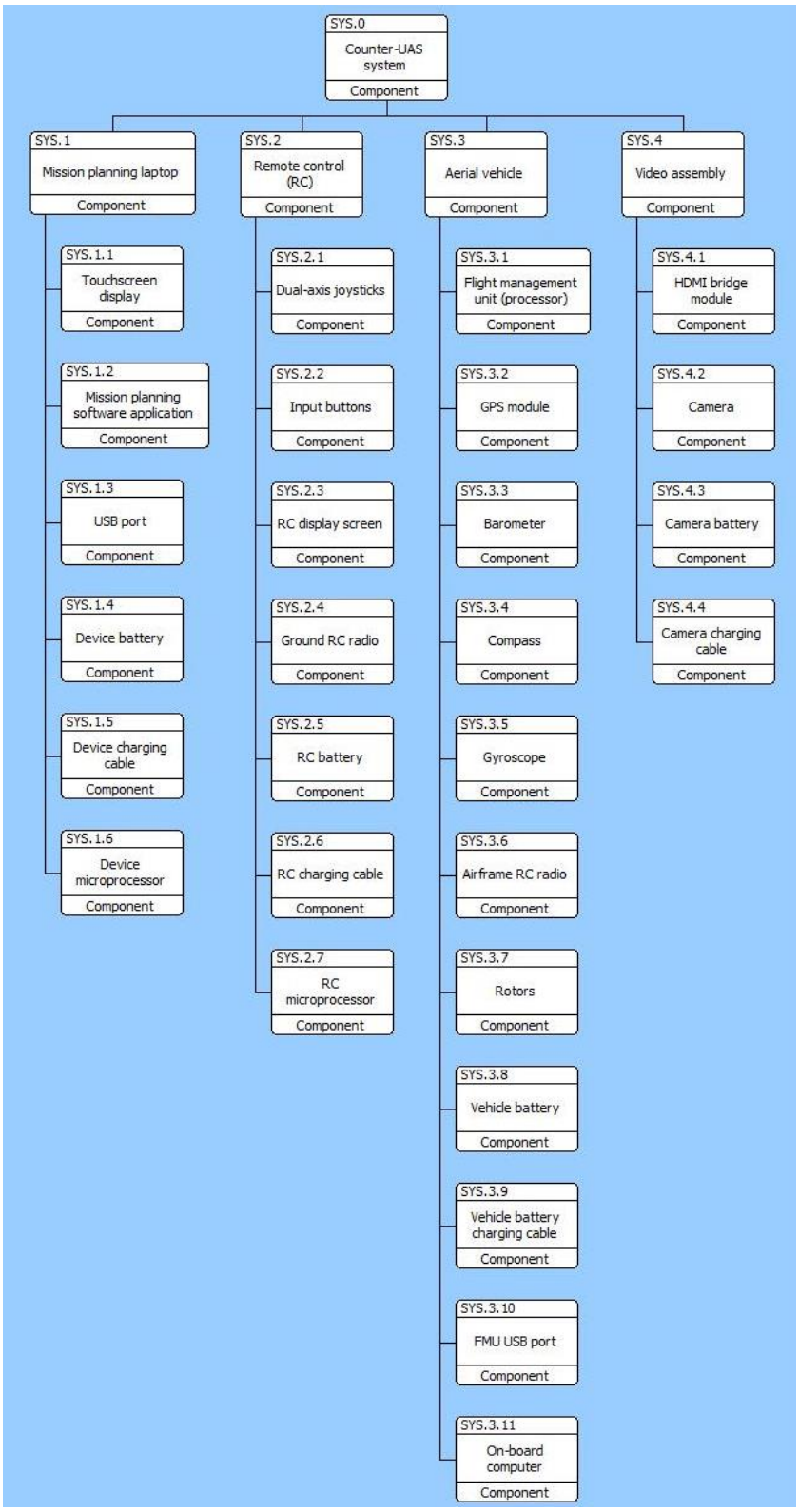


Figure 19. Physical Hierarchy Diagram

The systems view (SV) SV-1 diagram in Figure 20 shows the four top-level components in the system—namely the mission planning laptop, remote control, aerial vehicle, and video assembly—as well as the interfaces between these components.

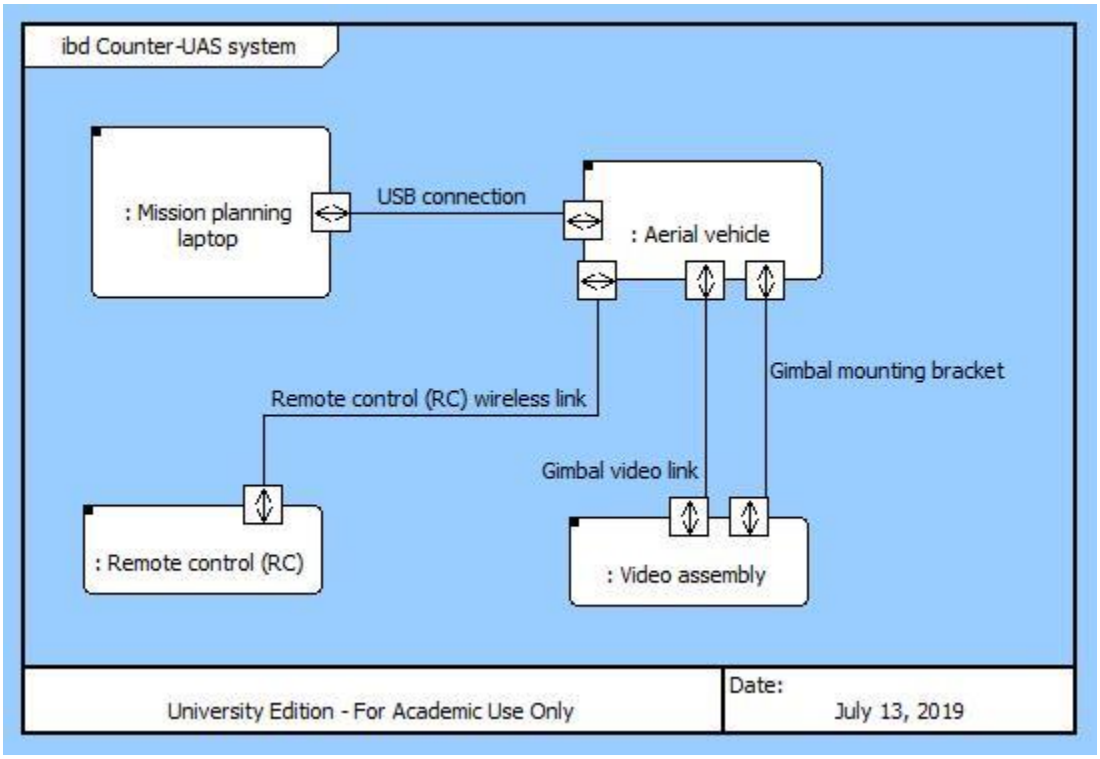


Figure 20. SV-1 Diagram for UAS

The SV-1 diagrams for each of the top-level components are illustrated in Figures 21–24. Figures 21 and 22 show that the touchscreen and the remote control's joysticks and buttons constitute the human-system interface.

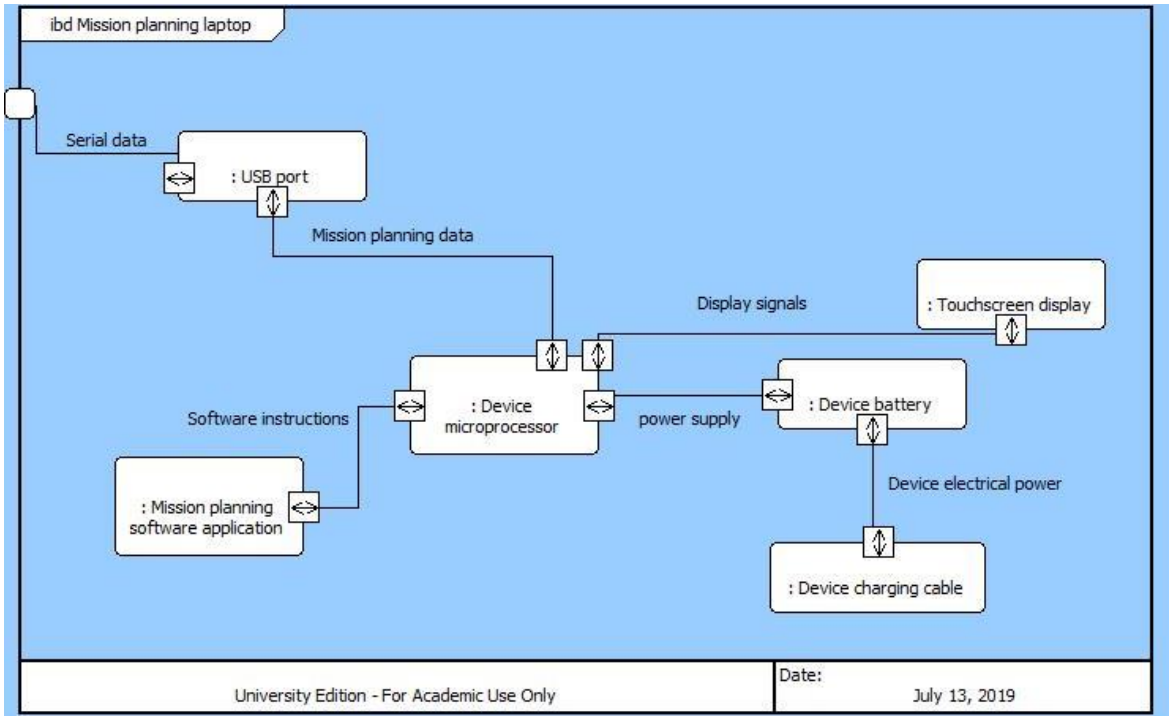


Figure 21. SV-1 Diagram for Mission Planning Laptop

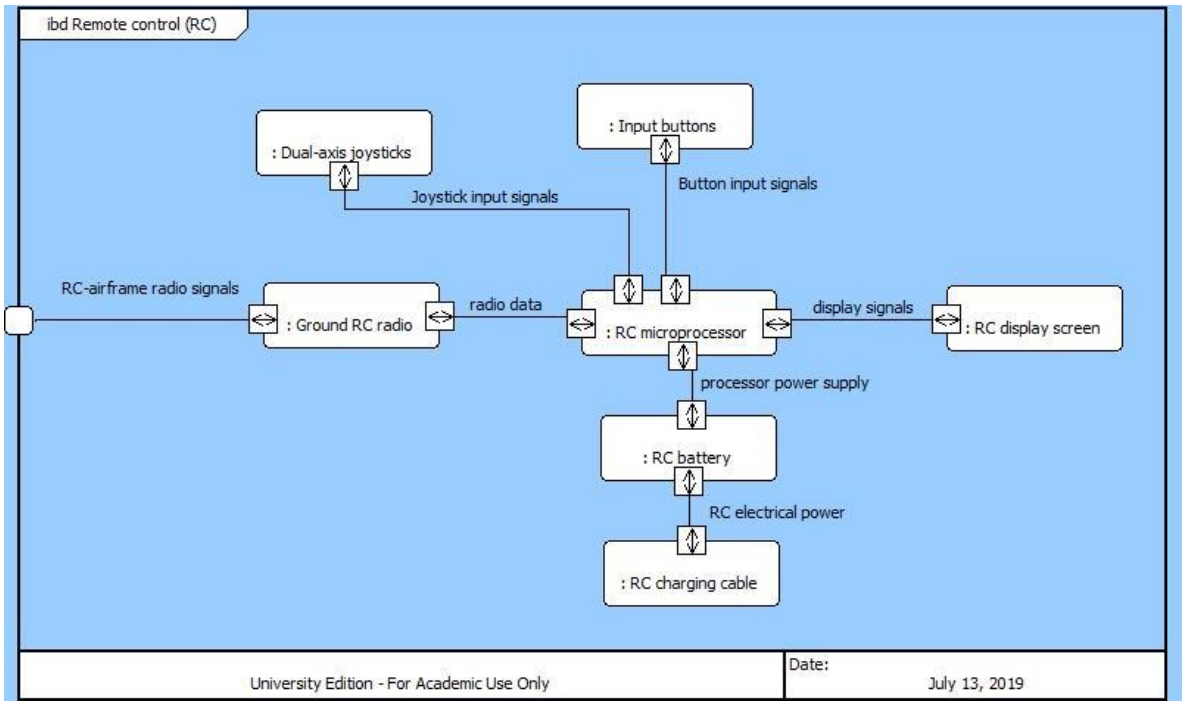


Figure 22. SV-1 Diagram for Remote Control

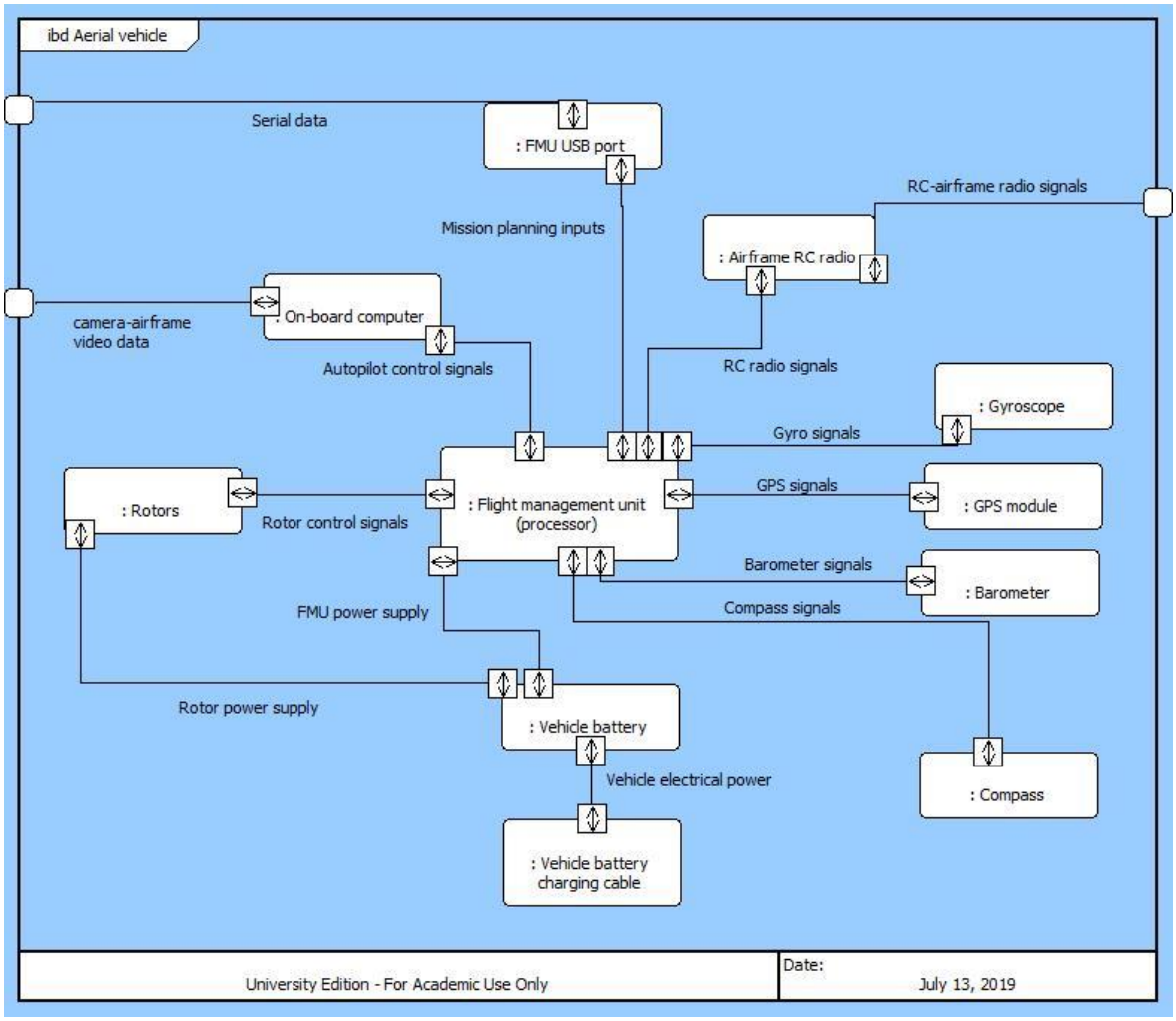


Figure 23. SV-1 Diagram for Aerial Vehicle

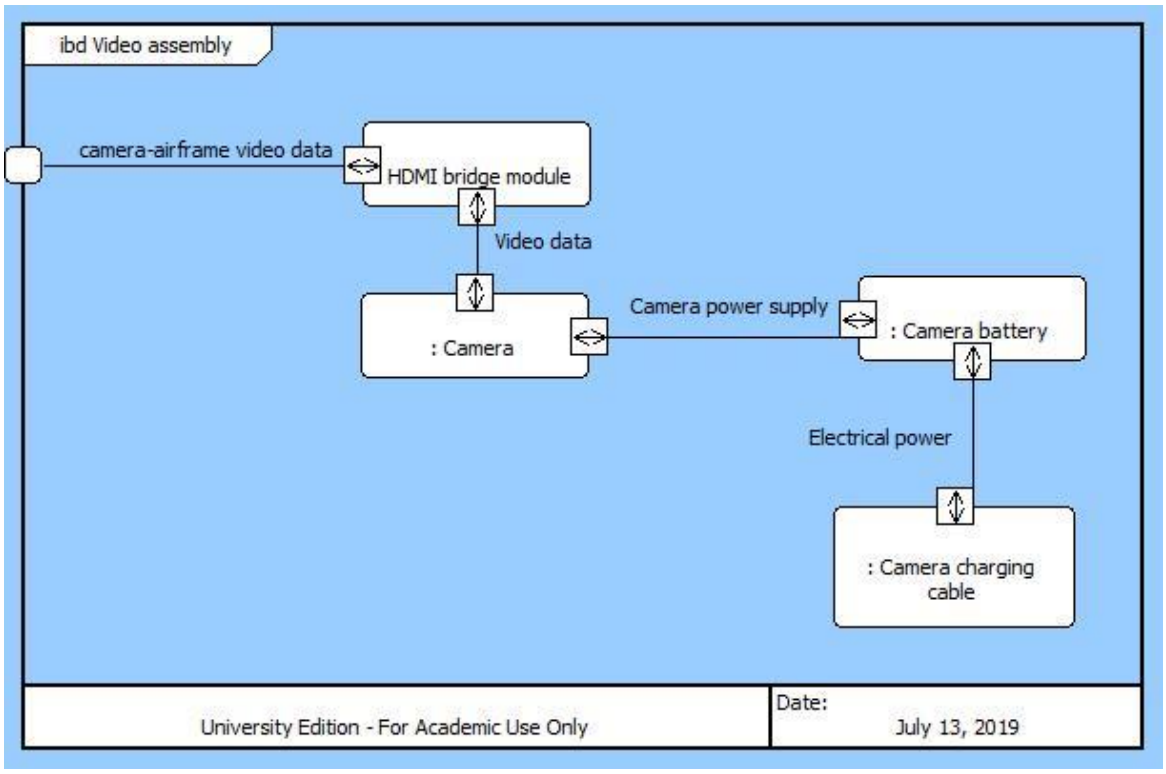


Figure 24. SV-1 Diagram for Video Assembly

The system boundary is overlaid on the OV-1 diagram in Figure 25, indicated by the red dashed line. Major inputs to the system include the following: spares (material), electricity for charging batteries (energy), mission-planning parameters (information), and sensors (GPS, compass, barometer) inputs from the environment (information). Major output from the system would be in the form of the recorded video footage (information).

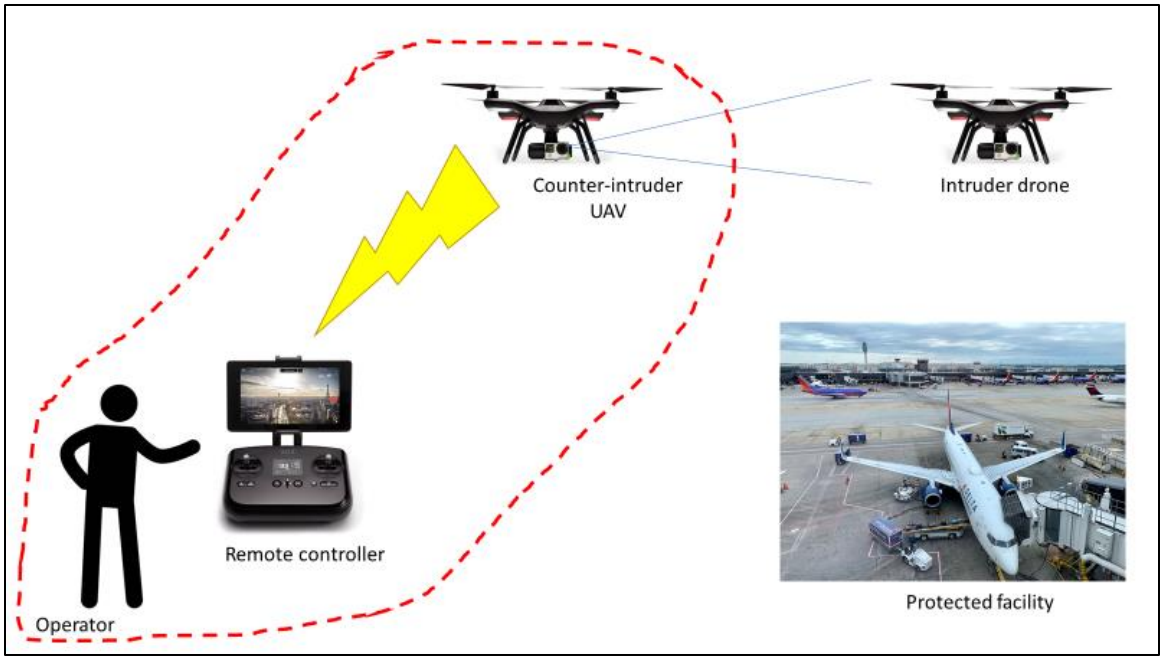


Figure 25. System Boundary. Adapted from 3D Robotics (n.d.); 3D Robotics (2015); Murph (2019).

THIS PAGE INTENTIONALLY LEFT BLANK

III. DEVELOPMENT OF CV ALGORITHM

This chapter describes the assumptions and key steps of the CV algorithm. One of the key functions of the counter-UAS system is to detect the presence of an airborne target in a field environment. The EO sensor in the camera produces a stream of images (i.e., video), which is then processed by the CV algorithm code running in the on-board computer.

A. DETECTION APPROACH, ASSUMPTIONS, AND TOOLS

As the problem formulation section in Chapter I stated, this thesis uses a feature-based object detection and localization CV algorithm to detect the target in a stream of images and localize its position. Given that the counter-UAS aerial vehicle has limited SWaP and needs to pursue the target in real time, a CV algorithm with lower computational demand is selected for implementation in this thesis.

The baseline scenario considers only the detection of a single airborne target against a sky background of largely homogeneous color (for example, blue, grey or white). These simplifying assumptions allow the thesis to focus on the range and angular performance of a COTS-based camera. Modifications to the baseline scenario, such as the detection of multiple airborne targets and detection of targets against non-homogenous backgrounds, are left as areas for future research.

The CV algorithm is first developed and tested in the MATLAB software environment, as this software has a rich array of image processing and visualization tools. In this thesis, the author made use of the Image Processing and Computer Vision Toolboxes in MATLAB which facilitates the use of various image processing, filtering, color-space segmentation and morphological operations to successfully detect and localize a target in the image frame.

B. ALGORITHM WORKFLOW

The sequential workflow of such operations in MATLAB forms the CV algorithm and the key tasks are broken down into the following steps:

1. Read the video file.

2. Read the current image frame.

3. Convert the image frame separately into Hue-Saturation-Value (HSV) color-space and grayscale images.

4. Extract the single layer array of hue values from the HSV image.

5. Create an image mask based on a threshold for hue values. This step exploits the distinct differences in hue values between the sky and the ground and the mask (based on hue threshold) helps to eliminate the surface foreground in the image frame.

6. Apply morphological erosion operation with a disk-shaped structuring element to improve the image mask by removing small remaining patches of the foreground left over from the thresholding operation.

7. Apply the image mask to the grayscale image.

8. Apply edge detection function with the Sobel method in MATLAB to the masked grayscale image to detect the edges of the target blob region in the sky.

9. Apply a morphological close operation with a line-shaped structuring element to the resulting image to form a target blob region with closed edges.

10. Perform two-dimensional median filtering on the resulting image twice to remove unwanted horizontal lines leftover from the edge detection step. This helps to refine the resulting image for target blob detection.

11. Apply a morphological dilate operation with a disk-shaped structuring element to the resulting image to fill up the target blob region. This forms an image mask to isolate the target.

12. Apply the image mask from step 11 to the masked grayscale image from step 7.

13. Apply the regionprops() function in MATLAB to the resulting image from step 12 to obtain the bounding box coordinates for the target.

14. Overlay the bounding box on the original image frame for visualization purpose.

15. Apply the detectMSERFeatures() function in MATLAB to the masked grayscale image from step 7 to detect the salient MSER features on the target. This serves as an alternative and backup means to extract the target location within the image frame. (This method involves fewer processing steps compared to the color-space segmentation and morphological operations between steps 8 to 13. The drawback, however, is that it only indicates the target location, but not the size of the target image blob which is needed for range estimation).

16. Repeat the entire workflow for the next frame in the video stream.

The preceding CV algorithm for object detection and localization represents the final product after many iterations of trying out different combinations of image processing steps and operations. The values selected for thresholding, median filtering and morphological operations were empirically determined based on the test videos used for the CV algorithm development. The experimental setup for obtaining the test videos are described in Chapter V. Refer to Appendix A for the MATLAB code.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CALIBRATION FOR VISUAL RANGE MEASUREMENT

This chapter describes the optical calibration process used to estimate physical dimensions from pixel measurements. The CV algorithm described in the preceding chapter is designed to localize the target's position within the image frame with the aid of a bounding box. Besides target localization, the dimensions of the bounding box—expressed in pixels—also serves as a proxy for the size of the target perceived by the EO sensor of the camera. This information can be exploited to estimate the range of the target from the camera by using the concept of ground sample distance (GSD) with a simplifying assumption and visual calibration on the camera with an object of known size at a fixed distance. Range estimation is necessary for the counter-UAS system to have complete spatial awareness of the target's relative position, and subsequently develop a flight path to intercept the target.

A. GROUND SAMPLE DISTANCE

In photogrammetry applications, a camera is typically used to survey and capture photographs of the ground terrain at a pre-determined altitude. Each pixel has a finite instantaneous field of view that is subtended by a portion of the ground being surveyed.

The GSD is defined as the distance between the projected centers of two consecutive pixels on the ground. The parameters that affect the GSD are the sensor width S_W (in mm), the focal length F_R(in mm), the distance from the image, or flight height H (in m), and the image width D_W (in m). See Figure 26.

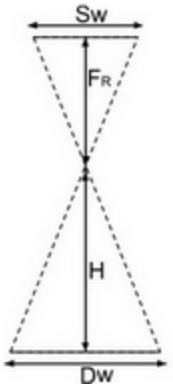


Figure 26. Parameters Affecting GSD. Source: Pix4D (n.d.).

The mathematical relationship between these parameters is defined in the following Equation (1):

$$\frac{H}{F_R} = \frac{D_W}{S_W}$$

Given a horizontal arrangement with a camera observing a target at a distance on the same plane, the same relationship holds true if D_w is substituted for the target dimension T_d, H is substituted for target range R, and Sw is substituted for size of target image T_i—expressed in pixels—perceived by the EO sensor. See Equation (2).

$$\frac{target\ range, R}{F_R} = \frac{target\ dimension, T_d}{target\ image\ size\ (pixels), T_i}$$

Re-arranging Equation (2) to express target range R in terms of other parameters, one easily observes that the target range R and the size of target image T_i share an inversely proportional relationship for a fixed target dimension T_d. Refer to Equation (3).

$$target\ range, R = \frac{F_R\ x\ target\ dimension, T_d}{target\ image\ size\ (pixels), T_i}$$

$$= \frac{k}{target\ image\ size\ (pixels), T_i}$$

The simplifying assumption made here is that the size of the target dimension is taken as a constant value for the purpose of analysis in this thesis. This is because the actual

size of the threat varies within a relatively narrow range of values (as outlined in Chapter I) for the class of drones under consideration.

B. EMPIRICAL ESTIMATION OF K

The constant k is empirically estimated by measuring the perceived pixel width of the image of a simple 10 cm x 10 cm black square on a white background in an indoor environment, at different distances from the camera, mounted on a tripod. A COTS-based camera, the GoPro Hero 4 Black, is used for the calibration, as it is the same payload that will be adopted for the counter-UAS system in this thesis. Videos of the black square were recorded at distances of 0.4 m, 0.7 m, 1.0 m, 1.3 m, 1.6 m, 1.9 m, and 2.2 m away from the camera (measured from the wall to the center of the tripod). The video was recorded at a video resolution of 1920 x 1080, and a frame rate of 30 fps in Linear field of view (FOV) mode. Frames at the original resolution were extracted from each video and the pixel width of the square in each frame was measured with the Paint image editing software application in Windows 10 environment. Figure 27 shows a sample photo at 0.4 m away from the camera.

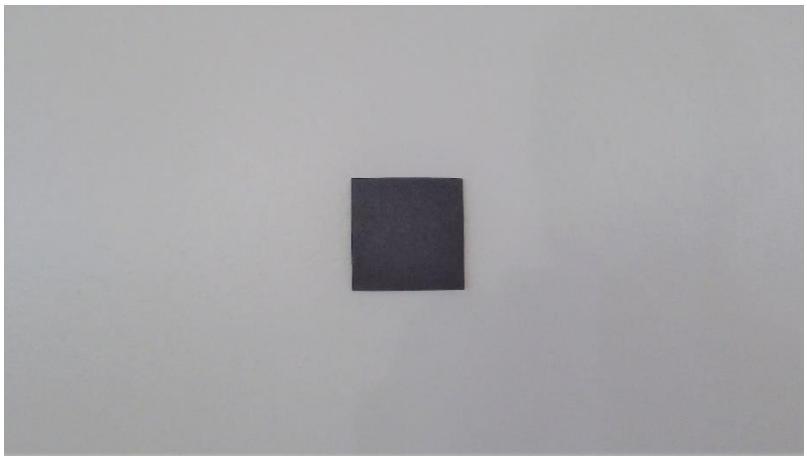


Figure 27. Sample Photo at 0.4 m Away from Camera

The pixel width measurements of the image were recorded at six different ranges. These data points were then fitted using the Excel statistical package assuming an inverse proportional model to obtain the constant k from this calibration process for a 10 cm wide

object. Table 1 summarizes the data for the calibration process, and Figure 28 shows the fitted graph that describes the inverse relationship between pixel width and distance to target.

Table 1. Empirical Variation of Pixel Width with Target Distance

object size: 10 cm		
pixel width (pixels)	1/pixel width	distance (meters)
270	0.00370	0.4
151	0.00662	0.7
106	0.00943	1
82	0.01220	1.3
68	0.01471	1.6
57	0.01754	1.9
49	0.02041	2.2

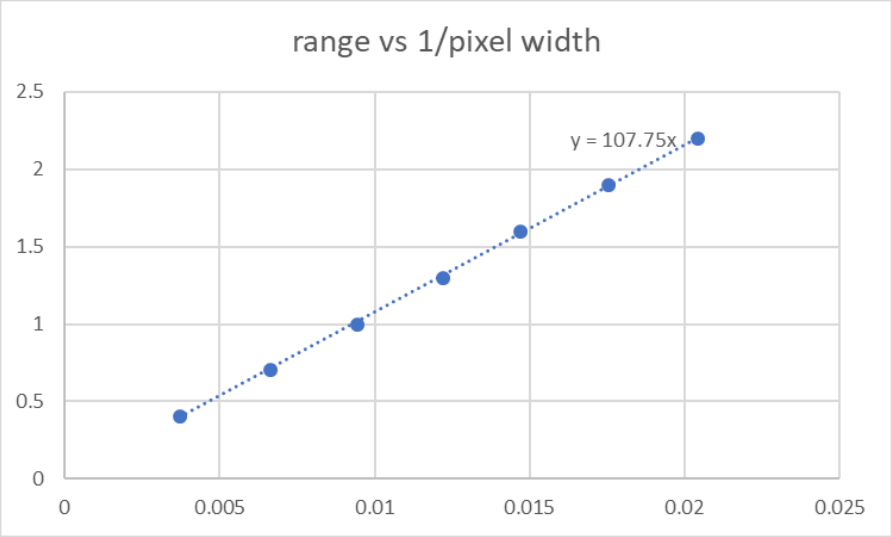


Figure 28. Fitted Graph of Inverse Relationship between Range and Pixel Width

The value of constant k obtained in this calibration process is 107.75. This constant was then multiplied by the appropriate factor of 3.3 to obtain the corresponding k constant, i.e., 355.575 for the nominal target (i.e., 33 cm for the width of the 3DR Solo UAS). The relationship between the pixel width and range to target for the nominal target at 1920 x 1080 video resolution is stated in Equation (4).

$$target\ range, R\ (meters) = \frac{355.575}{target\ image\ size\ (pixels), T_i}$$

THIS PAGE INTENTIONALLY LEFT BLANK

V. EXPERIMENTAL SETUP

This chapter describes the experimental setup to obtain the GPS-based measurements and corresponding CV-based estimates for range and angular evaluations. The experimental setup is geared toward obtaining videos from a hovering UAV (i.e., observer drone) with a camera payload observing another UAV (i.e., target drone) programmed with pre-determined maneuvers. The rationale for this setup is to realistically replicate the air-to-air encounter between the counter-UAS system and an intruder drone, and to use the perceived image frames from such encounters to develop the CV algorithm for object detection and localization. The observer drone hovers to maintain a relatively fixed observation position while the maneuvering drone functions as the target to test the range and angular accuracy of the visually estimated measurements.

A. RANGE MEASUREMENTS

Given the finite resolution of the camera's EO sensor, the range measurements aim to determine the empirical range limit of a COTS camera when operated together with the CV algorithm. Besides the camera's resolution limit, the CV algorithm based on MATLAB may also pose a limiting factor, and this appears in cases such as the minimum target blob size to trigger the construction of bounding boxes. In addition to this, in instances where bounding boxes are constructed for a detected target, the thesis aims to compare the accuracy of visually estimated measurements (i.e., range estimation from perceived pixel size with the actual range) with actual distances.

1. GPS Measurements

The actual distances are estimated by means of real-time kinematics (RTK) GPS receivers mounted on both the observer and target drones in the experiment. Compared to the conventional GPS receivers in cellphones, fitness trackers and watches, RTK GPS receivers offer significantly better position accuracy with centimeter-level precision. Besides the GPS receivers mounted on the drones, it is necessary to setup a GPS receiver base station on the ground to send correction signals to the drone-mounted receivers. The experiment made use of two Reach RTK Global Navigation Satellite System (GNSS)

receivers (drone-mounted units) and one Reach RS+ RTK GNSS receiver (base station) from EMLID to obtain and record GPS data. The receivers are capable of logging GPS data, which was extracted after the flights were completed. Refer to Figure 29 for a picture of the GPS measurement setup.

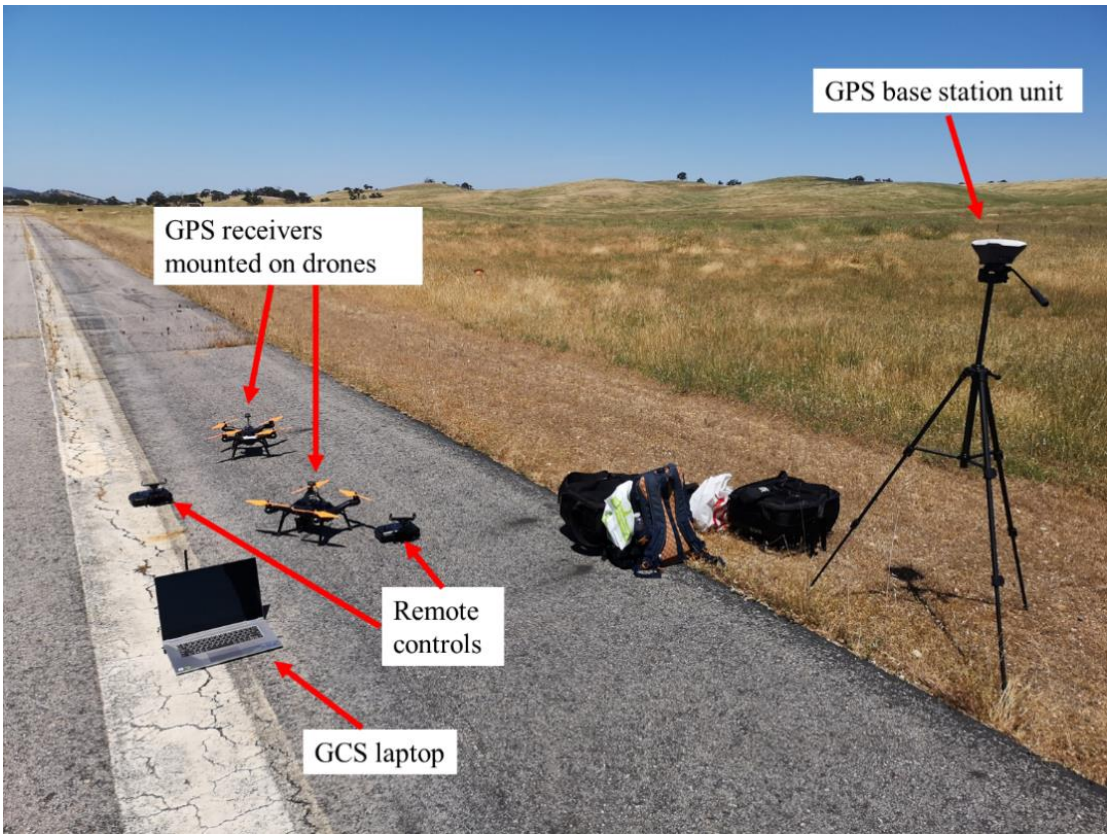


Figure 29. GPS Measurement Setup

2. Waypoint Measurements

The observer and target drones are flown by means of pre-programmed waypoints—essentially user-defined positions in three-dimensional space using altitude, latitude, and longitude coordinates—using the mission planning functions in the ground control station (GCS) software application. GCS software is

> typically a software application, running on a ground-based computer, that communicates with your UAV via wireless telemetry. It displays real-time data on the UAV's performance and position and can serve as a "virtual

cockpit," showing many of the same instruments that you would have if you were flying a real plane. A GCS can also be used to control a UAV in flight, uploading new mission commands and setting parameters. It is often also used to monitor the live video streams from a UAV's cameras. (ArduPilot Dev Team n.d.)

Flying the drones by waypoints has several benefits:

- It minimizes human errors in positioning the drones and makes the experiment repeatable.
- It automates the experiment task, improving efficiency and minimizes the human effort required to control two drones.
- The distance between waypoints can be measured in the mission planning software, and such measurements can be used to complement the GPS data.

The Windows-based Mission Planner GCS software application was chosen based on its ease of use, rich features, and compatibility with the drones used in this experiment. Refer to Figure 30 for a screenshot of the Mission Planner software interface.

Figure 30. Screenshot of Mission Planner Software

B. CHOICE OF DRONE

The 3DR Solo was selected as the platform for both the observer and target roles as it has dimensions and capabilities that are representative of the nominal threat. Furthermore, it can readily be mounted with a camera payload and is capable of streaming video in real time to the end user. This enables quick adjustments to be made in mid-air, and greatly streamlines the experiment workflow. Lastly, the drone is compatible with the Mission Planner GCS application, which helps to automate the drone flying.

C. CHOICE OF CAMERA PAYLOAD

The GoPro Hero 4 Black was selected as the camera payload for the experiment, as it is lightweight and compatible with the 3DR Solo drone platform. Additionally, this camera offers the Linear FOV video mode which helps to overcome the barrel distortion optical effect known as the "fish-eye" effect, which is common in wide-angle lens cameras. This feature is important to preserve accuracy in angular measurements. See Figure 31 for comparison of the distortion effect between the Standard FOV and the Linear FOV in the GoPro cameras.

Figure 31. Comparison of Barrel Distortion Effects.
Source: GoPro. (n.d.).

The linear FOV mode is only available for certain resolutions and frame rates (Coleman 2014). In the experiment, the video settings used are

- Resolution: 1080 p

- Dimensions: 1920 x 1080
- Aspect ratio: 16:9
- Frame rate: 30
- FOV mode: linear

D. DATA POINTS FOR RANGE MEASUREMENTS

The target drone is programmed to fly and hover for 5 sec at intervals of 10 m from the observer drone. Six data points were planned at distances of 10 m, 20 m, 30 m, 40 m, 50 m, and 60 m from the observer drone.

At each distance interval, the target is programmed by waypoint to hover at a fixed angle above the optical axis of the camera. This is done so that the target does not appear to rest on the horizon line in the camera's FOV—which complicates the detection task—but instead ensures that the target appears completely against the sky background for detection purpose. Figure 32 illustrates the differences.

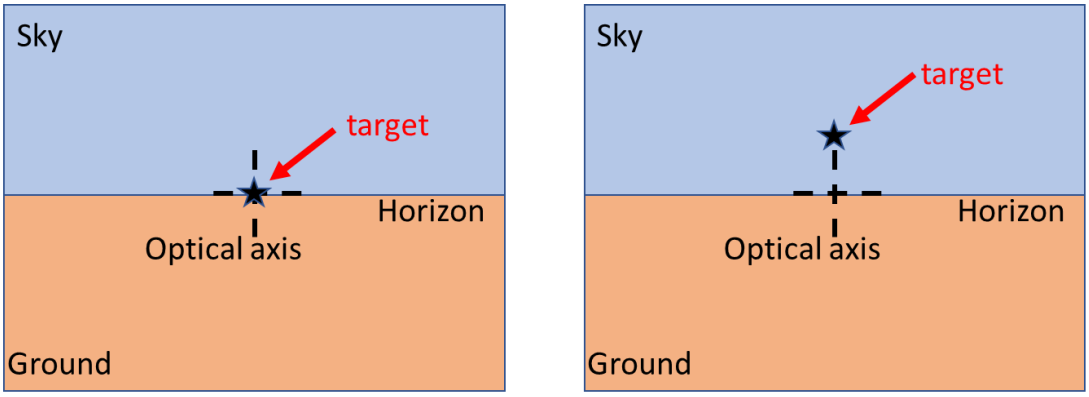


Figure 32. Comparison of Targets in Image Frame When Flying on Optical Axis and Off-Axis

For all distance intervals, the target drone was chosen to hover at different altitudes to maintain the same fixed angular displacement of 11.3^{o} in elevation from the optical axis (i.e., 11.3^{o} above the horizon). This angular displacement corresponds to a vertical

displacement of 2 m from the optical axis for every 10-m interval. The 2-meter vertical displacement was selected to buffer against position uncertainty as the drone has a 1-meter error when flown by waypoints. The level of precision is inferred from the input limitation of the Mission Planner application and is due to the limited precision of the navigation sensors onboard the 3DR Solo. Maintaining a fixed angular displacement from the optical axis (or above the horizon) helps to ensure the target blob appears approximately in the same spot in the image frame, minimizing the potential variation in optical distortion effects across different parts of the camera lens surface.

E. ANGULAR MEASUREMENTS

The EO sensor of the camera yields a two-dimensional output, i.e., the projection of the target within the camera's FOV, onto the image plane. The center of the image plane represents the optical axis of the camera. When the range of separation is known, the vertical and horizontal offset of the target position from the center of the image plane can yield information on the elevation and azimuth angular deviation of the line-of-sight (between target and observer) from the optical axis of the camera lens. This information can be further utilized as input signals in a control scheme to guide the observer drone to home in on the target.

F. OPTICAL DISTORTION EFFECTS

In the visual spectrum, the light rays from the target are modulated by the lens of the camera before impinging on the EO sensor to produce the image frame. In an ideal lens, the image produced is rectilinear. Imperfections in the optical design of a lens, however, can give rise to optical distortion effects. There are two main forms of optical distortion effects: barrel distortion and pincushion distortion. Barrel distortion can make straight lines in the image appear to bulge towards the edge of the image. Pincushion distortion has the opposite effect, and causes straight lines to appear to bend towards the center of the image. Figure 33 shows examples of the two distortion effects. Such effects may contribute to errors in the camera's spatial sensing of the target, and these effects can potentially affect the performance of the observer drone in its pursuit of the target. This study aims to characterize the accuracy of the angular measurements using a COTS-based camera.

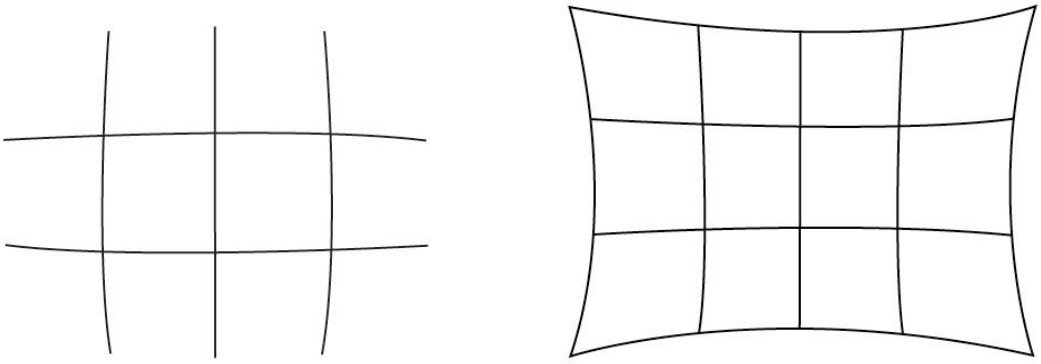


Figure 33. Examples of Optical Distortion Effects. Source: Petersen. (2016).

G. ESTIMATING ANGULAR DEVIATIONS

Figure 34 shows a nominal image frame where the detected target has been enclosed by a bounding box after image processing by the CV algorithm. The vertical and horizontal offset distances (in pixels) from the camera's optical axis can be easily determined through arithmetic operations between the pixel coordinates of the bounding box centroid and the center of image frame. The width of the bounding box is first used to estimate the range and to determine the pixel-distance relationship (i.e., distance represented by one pixel in the target plane), and thereafter the vertical and horizontal angular deviations can be calculated through trigonometry (assuming a rectilinear image). These estimates are compared against GPS-based data to evaluate the accuracy of the angular measurements.

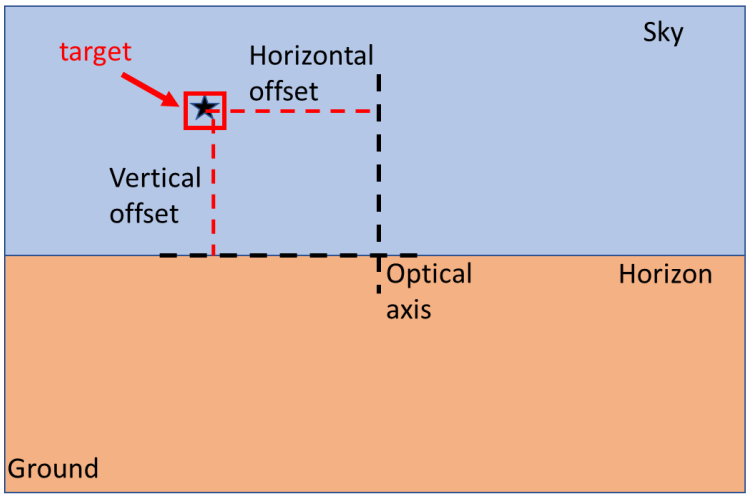


Figure 34. Definition of Vertical and Horizontal Offsets in the Image Frame

H. DATA POINTS FOR ANGULAR MEASUREMENTS

The target drone is programmed to fly and hover at different points in a single vertical plane at a pre-determined range (i.e., 20 m at Impossible City, 20 m and 10 m at Camp Roberts) from the observer drone. This results in different target positions across the image frame, as shown in Figure 35. Given the CV algorithm was designed for target detection above the observer's horizon; only target positions that met this criteria were analyzed for angular accuracy. The symmetrical design of the circular camera lens allows for the simplifying assumption that the bottom half of the lens surface can be approximated by the upper half. Nine data points across the image plane were planned for analysis.

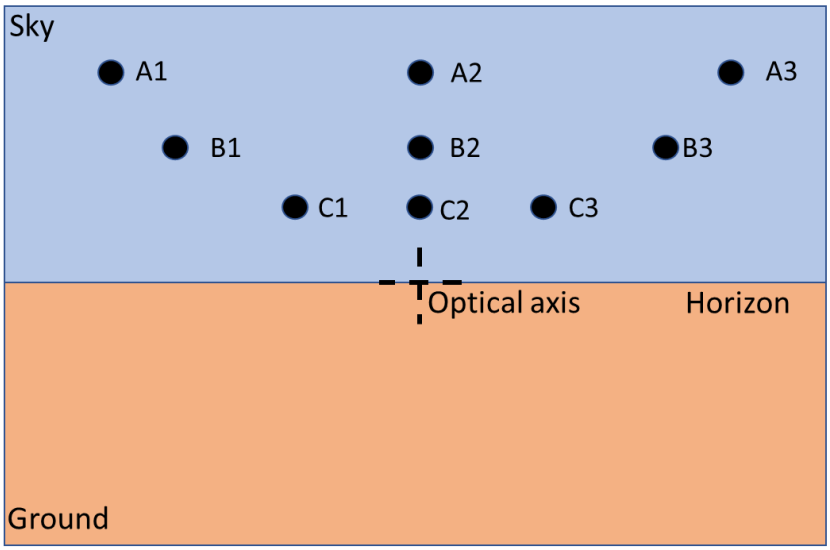


Figure 35. Illustration of Different Target Positions across Image Frame

The target drone is programmed by waypoints to hover at different positions in a plane that is perpendicular to the optical axis of the camera. The GoPro Hero4 Black camera has a vertical FOV of 55^{o} and a horizontal FOV of 94.4^{o} (GoPro n.d.) in the Medium FOV setting. These capabilities translate into theoretical offset limits of 10.41 m and 21.60 m in the vertical and horizontal dimensions, respectively. It should be noted here that GoPro did not publish the FOV details of the Linear FOV setting; however, visual observation of the camera playback shows that the Medium FOV setting has a narrower FOV than the Linear mode. Hence, the Medium FOV information was used to determine the angular boundaries of the waypoints.

The waypoints were programmed with different vertical and horizontal distances offset from the center of the plane. Figure 36 illustrates the definition of the spatial positioning of the drone within the target plane for video recording.

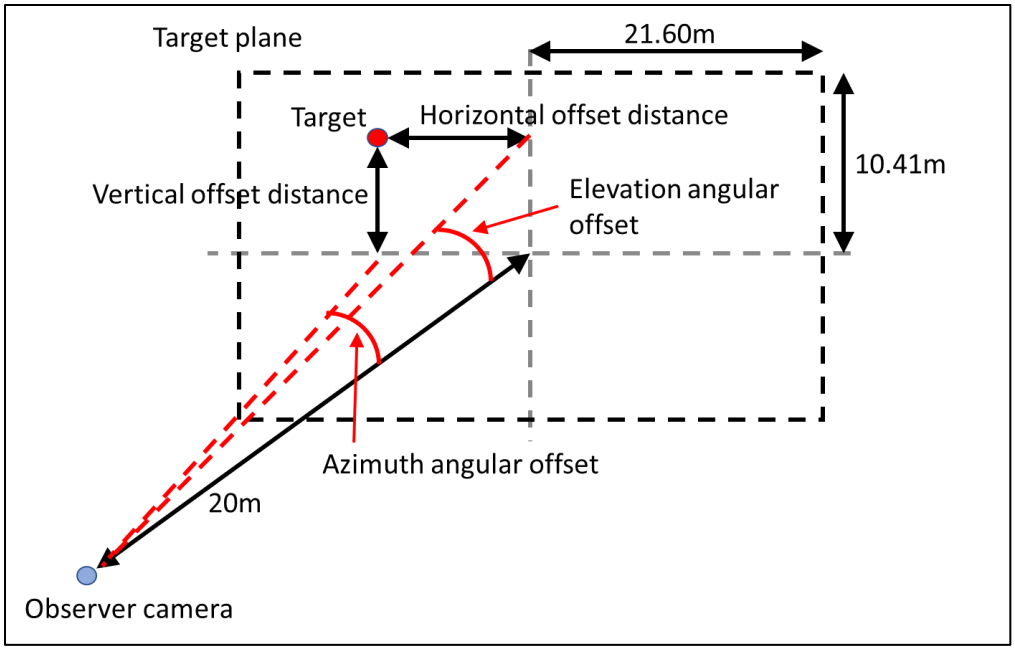


Figure 36. Definition of Spatial Positioning of Drone in the Target Plane

The nine target positions are labeled as indicated in Figure 35, and the respective offset distances for each position are summarized in Table 2. The direction of the offset is indicated by the sign of the value and follows the Cartesian coordinate convention with respect to the center of the target plane. Negative values for the horizontal offset distance and vertical offset distance indicate that the target is positioned to the left of the vertical axis and below the horizontal axis, respectively. The horizontal and vertical angular offsets are calculated from the range and the corresponding offset distances. The same sign convention applies to the angular values as well.

Table 2. Summary of Spatial Positioning for Different Target Positions

Target Position	Horizontal offset distance (meters)	Vertical offset distance (meters)	Azimuth angular offset (degrees)	Elevation angular offset (degrees)
A1	-18	9	-41.99	24.23
A2	0	9	0	24.23
A3	18	9	41.99	24.23
B1	-12	6	-30.96	16.70
B2	0	6	0	16.70
B3	12	6	30.96	16.70
C1	-6	3	-16.70	8.53
C2	0	3	0	8.53
C3	6	3	16.70	8.53

THIS PAGE INTENTIONALLY LEFT BLANK

VI. EXPERIMENTAL RESULTS OF RANGE AND ANGULAR MEASUREMENTS

This chapter analyzes the results of the range and angular measurements, and discusses alternative means to improve the tracking accuracy.

A. RANGE MEASUREMENTS

The first set of range measurements were conducted on 26 April 2019 at Impossible City, a Military Operations on Urban Terrain (MOUT) site within Fort Ord, in Monterey, California. The range measurements for the six different data points, based on the planned waypoint, GPS data, and visual estimation are summarized in Table 3. Bounding boxes were found only for the first two waypoints, R1 and R2; hence, the visual estimation technique for range was not possible beyond 20 m from the camera's position. The estimate error measures the difference between the CV estimated range and the planned distance as a percentage of the latter. Table 3 shows that the CV range estimation technique underestimates the actual distance by 46–48%. Correspondingly, this implies that the bounding box width is larger than expected given the inverse relationship between pixel width and target range. Further, as shown in Table 3, the GPS estimated ranges are fairly close to the planned waypoint distances, exhibiting a precision of approximately 1 m or less.

Table 3. Range Measurements at Impossible City.

Range Measurements at Impossible City. **Waypoint**	**Planned distance from waypoint to camera (meters)**	**Angular size (degrees)**	**GPS estimated range (meters)**	**Width of bounding box (pixels)***	**CV estimated range (meters)**	**Estimate error (%)**
R1	10	1.89	9.5	34	5.2	48%
R2	20	0.95	19.4	16	10.8	46%
R3	30	0.63	29.8	-	-	-
R4	40	0.47	40.1	-	-	-
R5	50	0.38	48.9	-	-	-
R6	60	0.32	59.2	-	-	-

*based on 960 x 540 resolution image frame

The time series plots showing variation in bounding box width and centroid coordinates over successive video frames are generated after processing the video recording of the waypoints with the CV algorithm in the MATLAB software environment. C_x refers to the x-coordinate of the centroid, while C_y refers to the corresponding y-coordinates. Figure 37 shows the raw data for variation in bounding box (BB) width over successive frames as well as the cleaned and detrended data over time. The data was cleaned by further post-processing the raw data to remove outliers. Likewise, Figure 38 shows the raw data for variation in centroid coordinates over successive frames. Figure 39 presents the cleaned data for the centroid coordinates. Analysis is done using cleaned data to minimize the effects of outliers.

The target hovers at each waypoint for eight seconds, and minimal variation is expected in the bounding box parameters in such instances. The plateau regions in the time series plots represent the occurrence of waypoints. This graphical approach is adopted to identify the occurrence of waypoints in the time series plots for all data.

'Flat-line' regions occur when bounding boxes disappear and the last known value of the parameter is stored in the time series variable until new values are written into the variable. Figure 37 clearly indicates two plateau regions representing the first two waypoints R1 and R2.

As the waypoints were planned to maintain a constant elevation angle from the optical axis, ideally both C_x and C_y would be expected to remain close to constant values. Figure 39 shows that the magnitude of variation in the values of C_x at waypoint R1 is approximately 50 pixels, and the corresponding magnitude of variation for C_y is approximately 13 pixels. It should be noted that the initial 1.5 seconds are not considered as the video shows the drone transiting into hovering position. At waypoint R2, the C_x variation magnitude decreases to approximately 31 pixels, while the C_y variation magnitude is approximately 23 pixels. A review of the video recording indicates that the observer drone platform is subject to environmental disturbances from wind when hovering. These environmental disturbances contributed to the observed rolling and pitching motion in the camera's recordings, and this caused variation in the centroid coordinates.

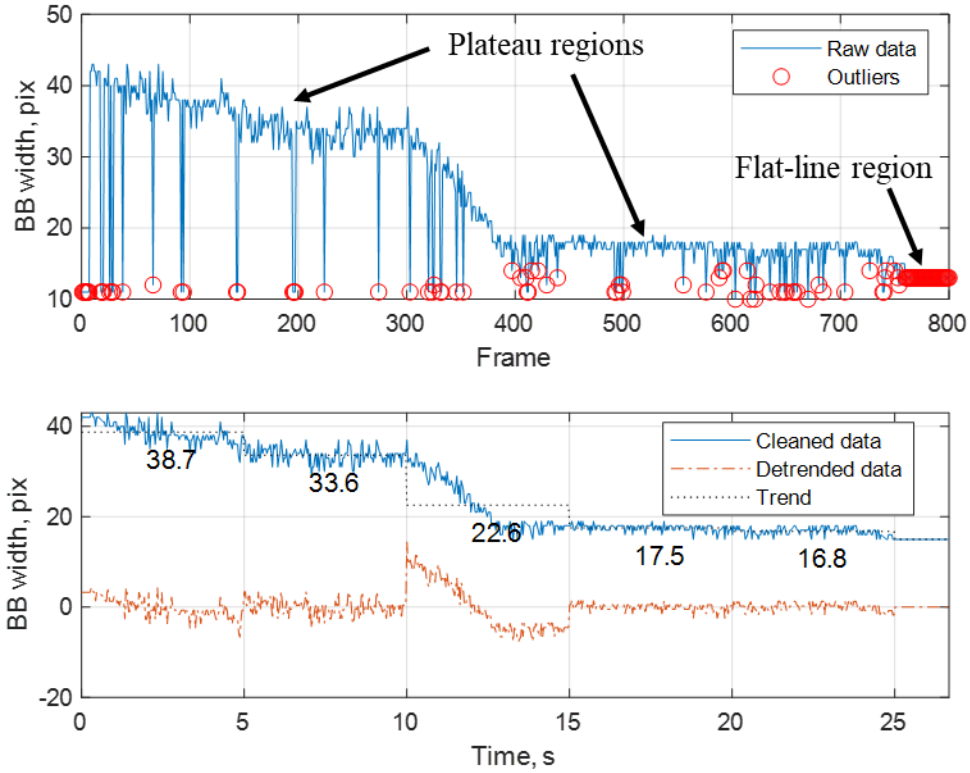


Figure 37. Variation in Bounding Box Width

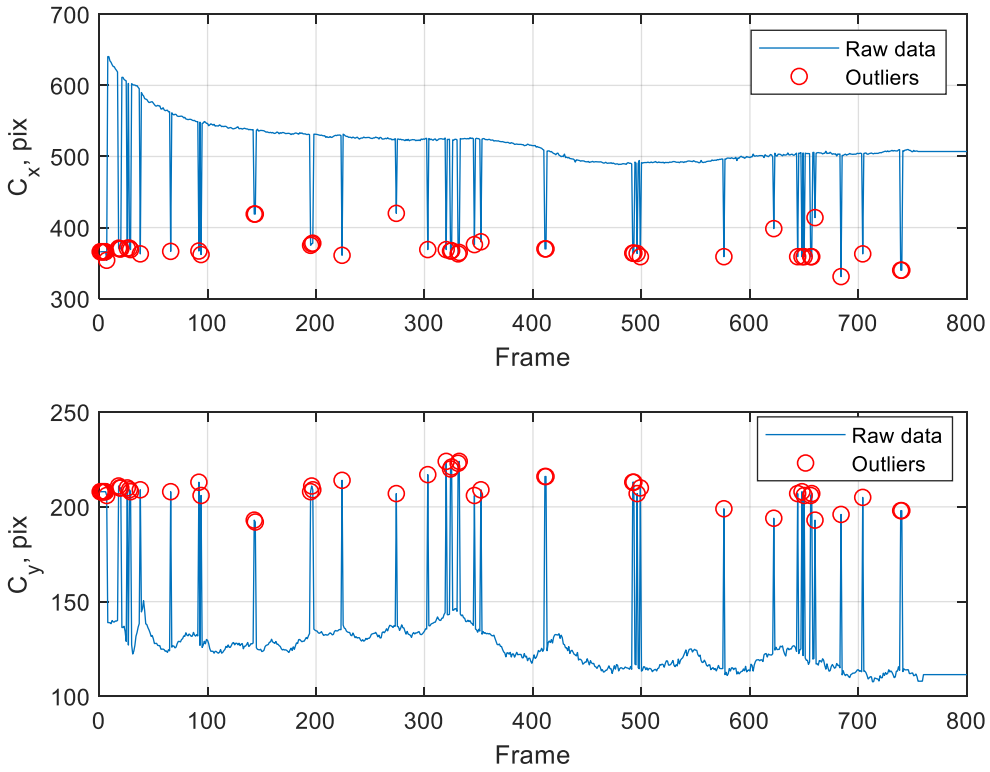


Figure 38. Outliers in Raw Data of Centroid Coordinates Variation

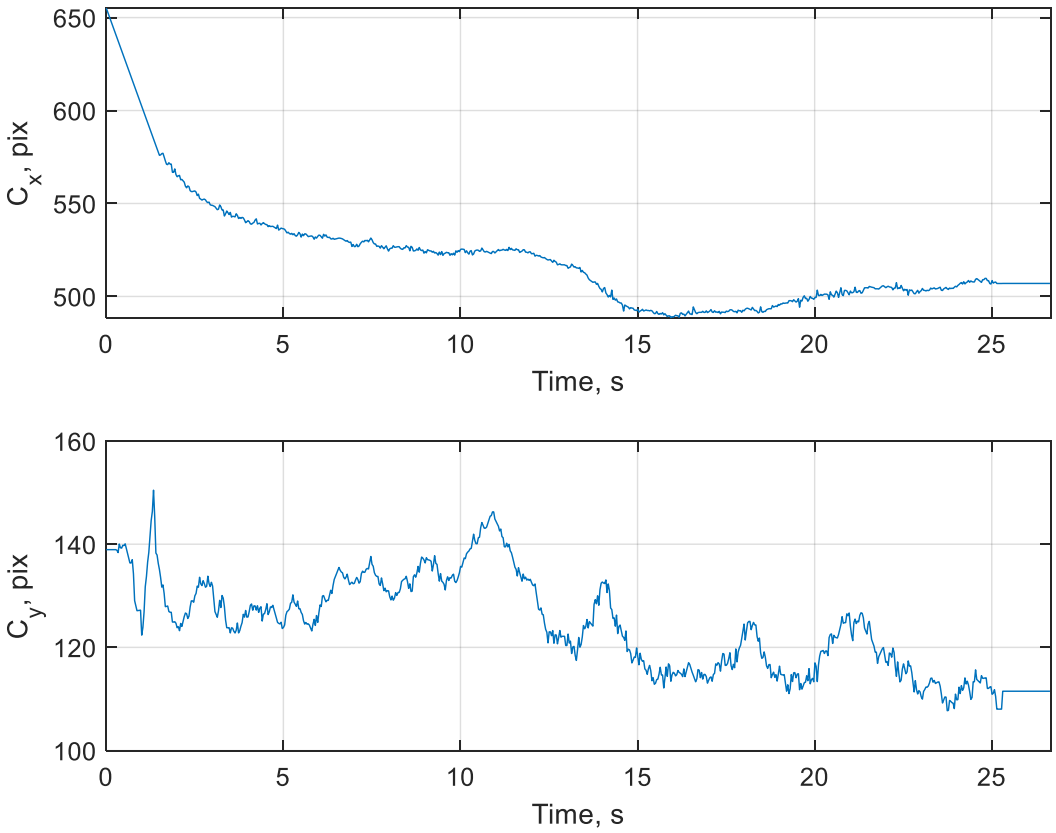


Figure 39. Post-processed Data (with Outliers Removed) Showing Centroid Coordinates Variation

A second set of range measurements was conducted in a similar manner at Camp Roberts located near Paso Robles, California, on 02 May 2019. The hovering duration for the range waypoints was adjusted to six seconds during this experiment in an effort to streamline the field test workflow. A challenge was encountered at Camp Roberts, as the CV algorithm was initially designed based on the landscape observed at Impossible City. The CV algorithm had to be adjusted in terms of the color-space threshold for both the hue and grayscale values in order to effectively perform color-space segmentation and localize the target within the image frame for videos recorded at Camp Roberts. As the landscape and sky conditions at Camp Roberts were more homogeneous in color tone compared to Impossible City, there were fewer morphological operations involved in the modified CV algorithm to remove unwanted image artifacts. Refer to Figure 40 and Figure 41 for a comparison of the landscape and sky conditions at the two field test sites. Implicitly, this

also highlighted the limitation of the CV algorithm, as it is not able to effectively adapt to different landscapes and cloud cover. The modified CV algorithm is appended in Appendix B for reference.

Figure 40. Snapshot of Landscape and Sky Condition at Impossible City

Figure 41. Snapshot of Landscape and Sky Condition at Camp Roberts

The target drone was programmed with waypoints maintaining the same elevation angle of 11.3º from the optical axis of the camera. The data points used for range measurement at Camp Roberts included the same distances as those used at Impossible City, but additional data points were added at 4 m increments between 0 m and 20 m. These were done to compare and verify the results gathered at Impossible City, and to gain further insight into the variation in pixel width within the useful bounding box detection range.

The results are summarized in Table 4. Figure 42 shows the raw data for variation in bounding box width over successive frames as well as the cleaned and detrended data over time. In this case, five plateau regions—corresponding to the first five waypoints—can be discerned. Figure 43 shows the raw data for variation in centroid coordinates over successive frames. Figure 44 presents the cleaned data for the centroid coordinates. Analysis is done using cleaned data to minimize the effects of outliers.

While the same GPS distance measurement setup was employed at Camp Roberts, no meaningful data could be extracted from the GPS logs. This problem affected all data points for range and angular measurements at Camp Roberts. After all the drone flights at Camp Roberts were completed, it was observed that the GPS antennas on the drones were

not perpendicular to the plastic antenna masts. The GPS antennas were designed to be mounted on flat surfaces and the plastic antenna mast was an improvised solution comprising 3D printed components that were joined together by strong adhesive and duct tape. It is postulated that mechanical shocks from repeated drone landings and in-flight vibrations may have contributed to the deterioration of the antenna mast joint over time. Refer to Figure 45 for a comparison of the mast joint condition. A loosening of the joint would have caused the GPS antenna to become tilted, and the quality of the GPS signals received would have been degraded (EMLID n.d.).

Table 4. Range Measurements at Camp Roberts

Waypoint	Planned distance from waypoint to camera (meters)	Angular size (degrees)	GPS estimated range (meters)	Width of bounding box (pixels)*	CV estimated range (meters)	Estimate error (%)
RR1	4	4.7	-	70	5.1	-27%
RR2	8	2.4	-	50	7.1	11%
RR3	12	1.6	-	44	8.1	33%
RR4	16	1.2	-	35	10.2	36%
RR5	20	1.0	-	22	16.1	20%
RR6	30	0.6	-	-	-	-
RR7	40	0.5	-	-	-	-
RR8	50	0.4	-	-	-	-
RR9	60	0.3	-	-	-	-

*Based on 1920 x 1080 resolution image frame

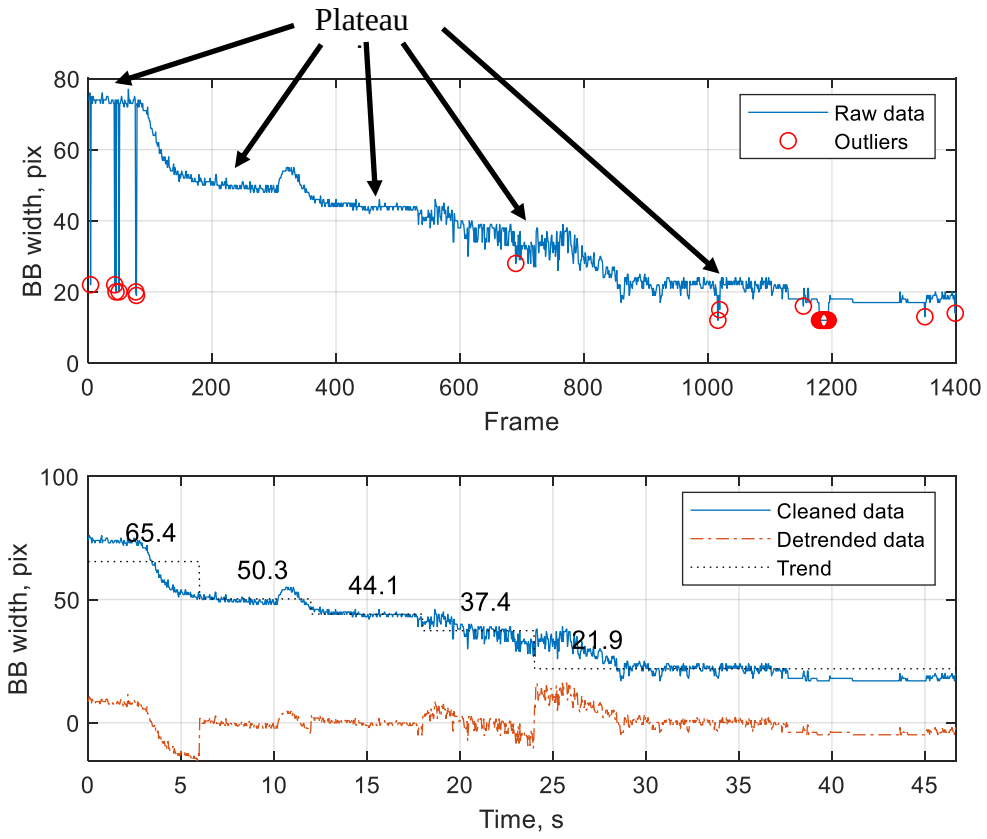


Figure 42. Variation in Bounding Box Width

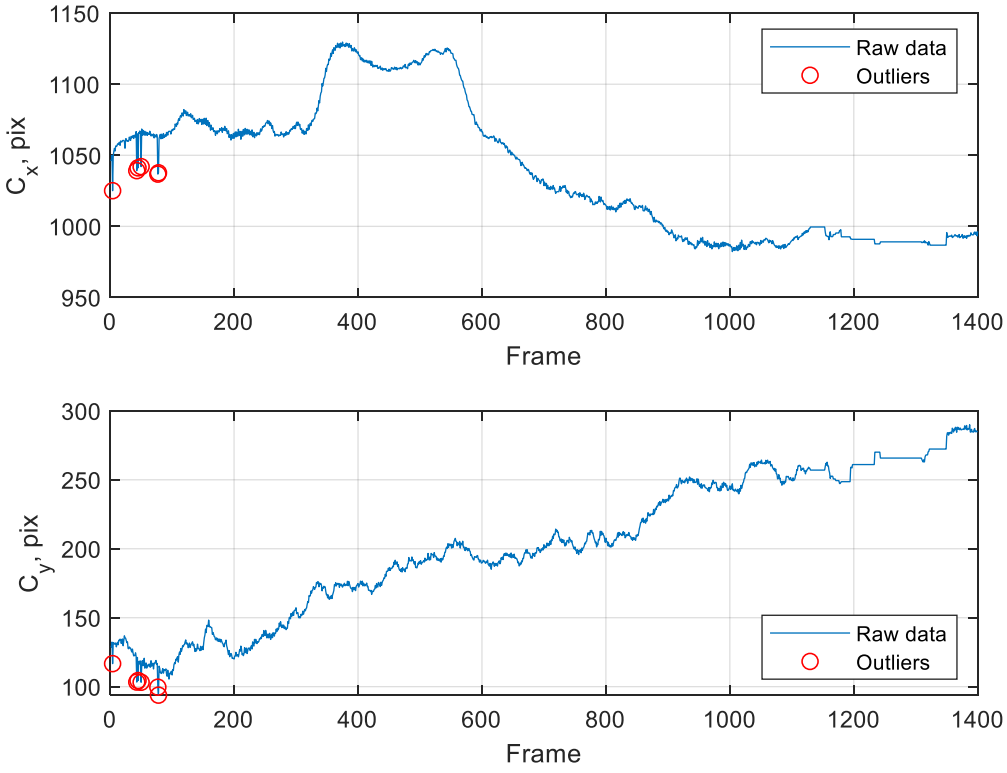


Figure 43. Outliers in Raw Data of Centroid Coordinates Variation

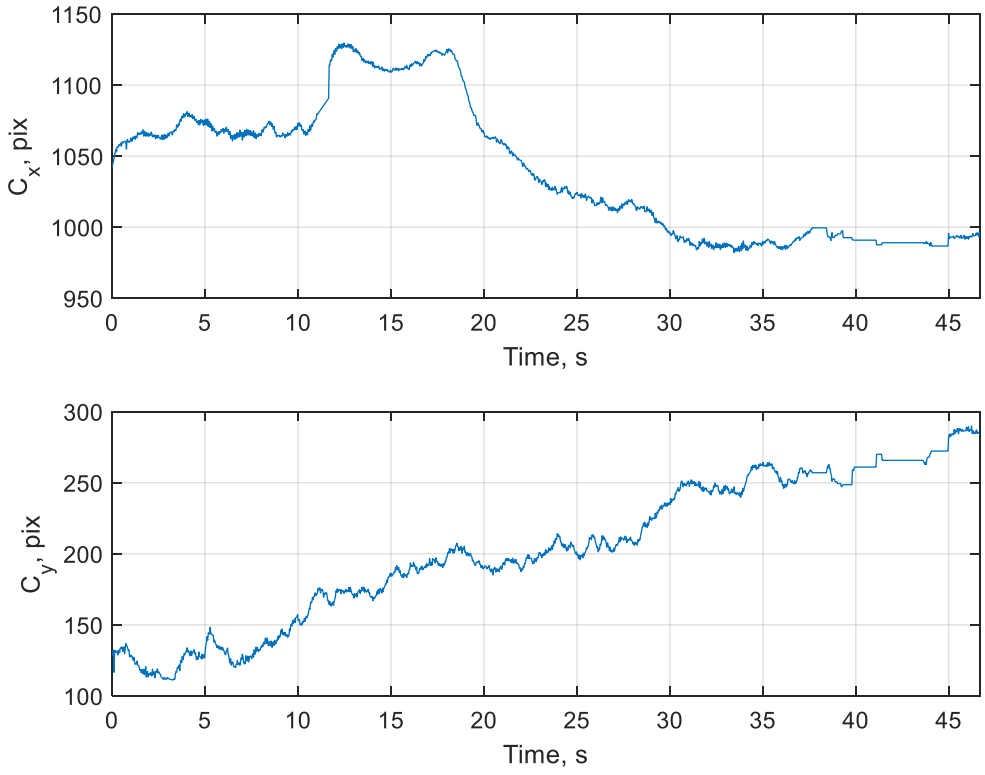


Figure 44. Post-processed Data (with Outliers Removed) Showing Centroid Coordinates Variation

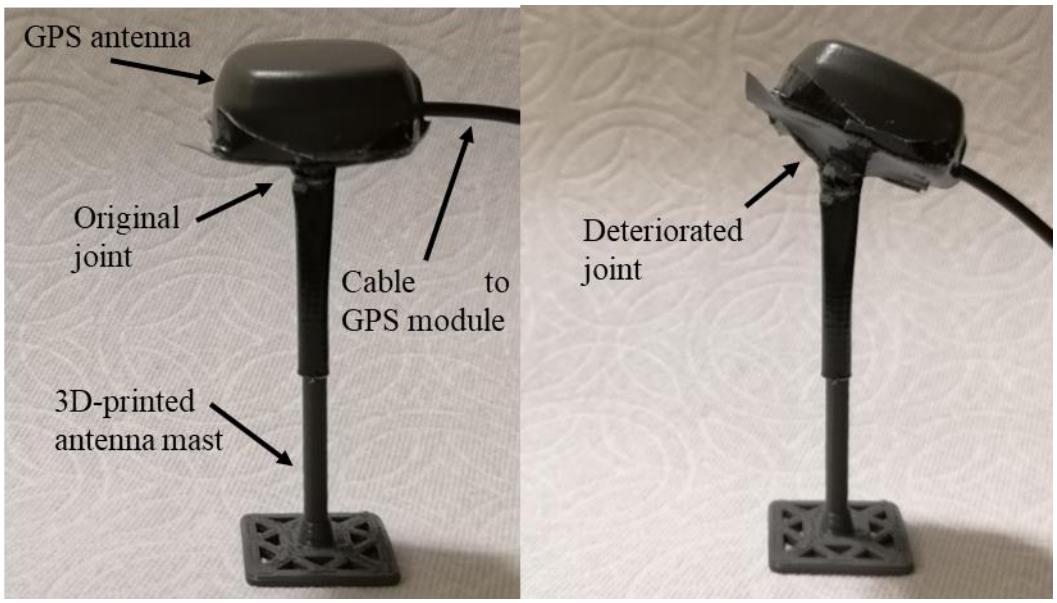


Figure 45. Comparison of Mast Joint Condition for Normal Joint (Left Image) and Deteriorated Joint (Right Image)

B. RANGE MEASUREMENT LIMITS AND ACCURACY

The results from Table 4 show that bounding boxes were found only for the first five waypoints RR1–RR5; hence, CV estimation technique for range was not possible beyond 20 m from the camera position. This verifies the same range limitation that was observed from the data in Table 2 and could form a baseline measure against which other algorithms could be compared.

In terms of accuracy, the range estimation technique does not perform well in estimating distance to the target, with large error percentages ranging from 11–36% in magnitude. There was no noticeable trend in the error percentages. In nearly all the data points, the algorithm also underestimates the actual distance to the target. The sole data point with range overestimation occurred at waypoint RR1, which was the position nearest to the camera. Given that the estimated range varies inversely with the perceived bounding box pixel width, the same magnitude in errors arising from target pixel width perception (for example, sensor pixel noise in resolving the target from the background) will contribute to much greater errors at shorter distances from the camera. This is a drawback of the range estimating relationship. Nevertheless, within the detectable range limits, tracking the *trends* in target pixel width could still function as a useful indicator of whether a target is *approaching* or *receding* from the observer position. This could be employed as a possible heuristic in a control algorithm to guide the observer drone to home in on the target.

The percentage error is also significantly greater when processing a lower resolution image frame. The data points in Table 3 were obtained from processing 960 x 540-pixel image frames and the percentage errors in range estimation all exceeded 45%. In comparison, the data points in Table 4 were based on 1920 x 1080-pixel image frames and the percentage errors were smaller, varying between 11% and 36% in magnitude. These results are within expectations, as higher resolution images frame can better represent finer details on the same given object.

In Figure 37, the flat line first occurs when the pixel width decreases to 13 pixels (while the lowest recorded pixel width in the time series plot was 11 pixels for a short

instance). By contrast, in Figure 42 the flat line first occurs at 17 pixels (while the lowest recorded pixel width in the time series plot was 12 pixels). Given that the image frames that generated the two time series plots in Figures 37 and 42 were of different resolutions, the proximity in values of the lowest recorded pixel width suggests the construction of bounding boxes might have an inherent lower bound arising from possible software limitation of the bounding box function. This observation could serve as a baseline to compare performance in bounding box construction with other image processing libraries such as OpenCV, as well as other types of object detection and localization algorithms, such as machine learning-based approaches.

The possible sources of error in the target dimension (based on bounding box width) could include sensor noise and morphological dilation operations in the CV algorithm. First, the sensor has to resolve the boundary between the target and the background in order to perceive the target. A practical EO sensor will have some finite noise in its output; hence, the perceived width of the target and consequently the width of the bounding box will fluctuate. Second, morphological dilation operations at the pixel level are necessary to recover parts of the target image blob, which may have been "erased" by prior morphological erosion operations targeted at other artifacts in the video frame. The precision of such morphological operations is limited and influenced by the choice of structuring element, which affects every pixel and its neighborhood. While dilation operations may help to recover the "erased" parts of the target blob, the additive nature of the operation also adds pixels to the boundaries of the unaffected portions of the target blob, and that can inadvertently increase the perceived width of the target and bounding box. This can contribute to range underestimation.

Furthermore, the magnitude of variation in the centroid coordinates is comparable to those of the data points from Impossible City. The video recording also shows observable rolling and pitching motion of the camera on board the observer drone platform at Camp Roberts. At waypoint RR3, the target drone was observed to marginally shift to the right when hovering, which accounts for the increase in the C_x value at that waypoint. It is postulated that the lateral shift could be due to momentary errors in the target drone's

GPS self-localization. When waypoint RR3 was completed, the target drone was observed to laterally shift to the left for subsequent waypoints.

The value of C_y was observed to trend upwards as the target drone flew farther away from the camera. Given that the vertical pixel coordinate convention of the image frame increases towards the bottom of the scene, this implies that the projection of the target drone on the image plane was approaching the center of the image plane as opposed to maintaining a constant elevation angle as planned initially. Consequently, this suggests a finite pitch angular error in the orientation of the camera.

C. ANGULAR MEASUREMENTS

At Impossible City, nine waypoints across the target plane, at 20 m from the camera position (measured from center of target plane), were planned for analysis. The relative positions of these waypoints are summarized in Figure 46 for ease of reference.

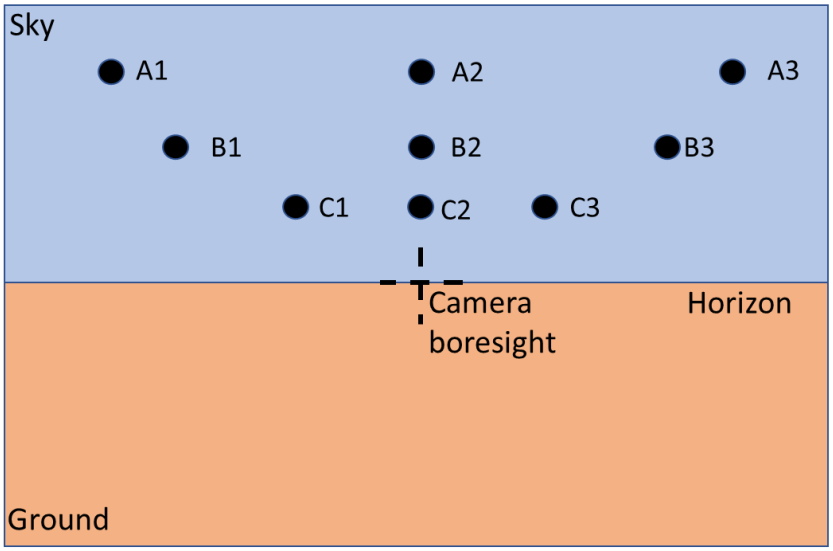


Figure 46. Illustration of Different Target Positions across Target Plane

The summary of the planned horizontal and vertical offset distances of the nine waypoints, as well as the theoretical resultant angular offsets in azimuth and elevation are shown in Table 5. At Camp Roberts, these nine waypoints were repeated to verify the

results from Impossible City. In addition, nine more waypoints (following similar geometrical arrangement) in a target plane 10 m from the camera position were flown. The horizontal and vertical offset distances were chosen to generate different angular offsets from the prior set of nine waypoints. The rationale is to populate the image plane with more data points for analysis. The summary of the planned spatial positioning for these nine waypoints is summarized in Table 6.

Table 5. Summary of Planned Spatial Positioning for Different Target Positions Observed at 20 m from Camera Position (Impossible City and Camp Roberts)

Target Position	**Horizontal offset distance (meters)**	**Vertical offset distance (meters)**	**Azimuth angular offset (degrees)**	**Elevation angular offset (degrees)**
A1	-18	9	-42.0	24.2
A2	0	9	0	24.2
A3	18	9	42.0	24.2
B1	-12	6	-31.0	16.7
B2	0	6	0	16.7
B3	12	6	31.0	16.7
C1	-6	3	-16.7	8.5
C2	0	3	0	8.5
C3	6	3	16.7	8.5

Table 6. Summary of Planned Spatial Positioning for Different Target Positions Observed at 10 m from Camera Position (Camp Roberts)

Target Position	**Horizontal offset distance (meters)**	**Vertical offset distance (meters)**	**Horizontal angular offset (degrees)**	**Vertical angular offset (degrees)**
AA1	-6	3	-31.0	16.7
AA2	0	3	0	16.7
AA3	6	3	31.0	16.7
BB1	-4	2	-21.8	11.3
BB2	0	2	0	11.3
BB3	4	2	21.8	11.3
CC1	-2	1	-11.3	5.7
CC2	0	1	0	5.7
CC3	2	1	11.3	5.7

A montage of the snapshots for the nine waypoints is shown in Figure 47. Table 7 summarizes the comparison of the GPS-based angular measurements, planned angular offsets, and CV-based angular estimation of the nine waypoints flown at Impossible City (observed at 20 m). Table 8 compares the errors between the GPS-based angular measurements, planned angular offsets, and CV-based angular estimation. Figure 48 shows the visualization of the azimuth and elevation errors between CV estimates and planned offset in Table 8 for the different waypoints.

Figure 47. Montage of Waypoints at Impossible City
(Observed at 20 m)

Table 7. Summary of Angular Measurements at Impossible City
(Observed at 20 m)

Positions	Azimuth (degrees)			Elevation (degrees)		
	Planned azimuth angular offset (degrees)	GPS-based measurements (degrees)	CV-based estimation (degrees)	Planned elevation angular offset (degrees)	GPS-based measurements (degrees)	CV-based estimation (degrees)
A3	42.0	42.3	-	24.2	26.5	-
A2	0.0	-5.4	-	24.2	24.1	-
A1	-42.0	-43.4	-25.1	24.2	23.0	15.2
B3	31.0	27.8	27.9	16.7	16.5	16.5
B2	0.0	-5.1	5.5	16.7	16.7	21.4
B1	-31.0	-33.9	-22.6	16.7	16.8	14.3
C3	16.7	14.5	24.4	8.5	7.3	11.7
C2	0.0	-5.4	7.0	8.5	8.1	11.4
C1	-16.7	-19.6	-6.8	8.5	7.4	7.2

Table 8. Summary of Angular Errors at Impossible City
(Observed at 20 m)

Positions	Azimuth (degrees)			Elevation (degrees)		
	Error between GPS and waypoint (degrees)	Error between CV estimate and waypoint (degrees)	Error between CV estimate and GPS (degrees)	Error between GPS and waypoint (degrees)	Error between CV estimate and waypoint (degrees)	Error between CV estimate and GPS (degrees)
A3	-0.3	-	-	-2.3	-	-
A2	5.4	-	-	0.1	-	-
A1	1.5	-16.9	-18.3	1.2	9.1	7.9
B3	3.1	3.1	-0.1	0.2	0.2	0.0
B2	5.1	-5.5	-10.6	0.0	-4.7	-4.7
B1	3.0	-8.4	-11.3	-0.1	2.4	2.5
C3	2.2	-7.7	-9.9	1.3	-3.2	-4.4
C2	5.4	-7.0	-12.4	0.5	-2.9	-3.3
C1	2.9	-9.9	-12.8	1.1	1.3	0.2

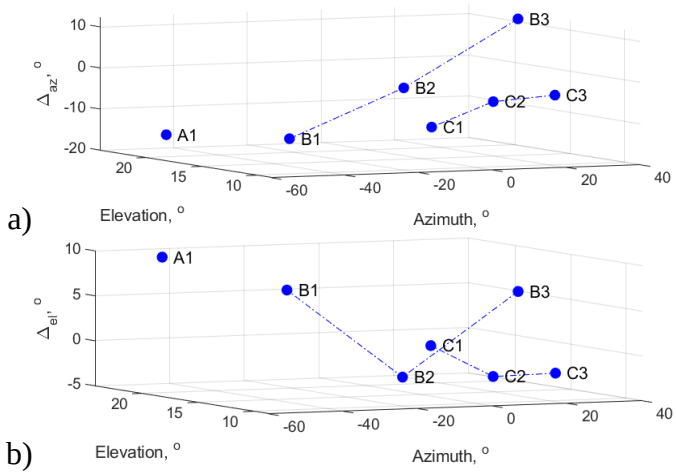


Figure 48. Visualization of Table 8 Data for (a) Azimuth Error and (b) Elevation Error

A montage of the snapshots for the same waypoints flown at Camp Roberts (observed at 20 m) is shown in Figure 49. Table 9 summarizes the data for these waypoints, while Table 10 compares the errors. Figure 50 shows the visualization of the azimuth and elevation errors between CV estimates and planned offset in Table 10 for the different waypoints.

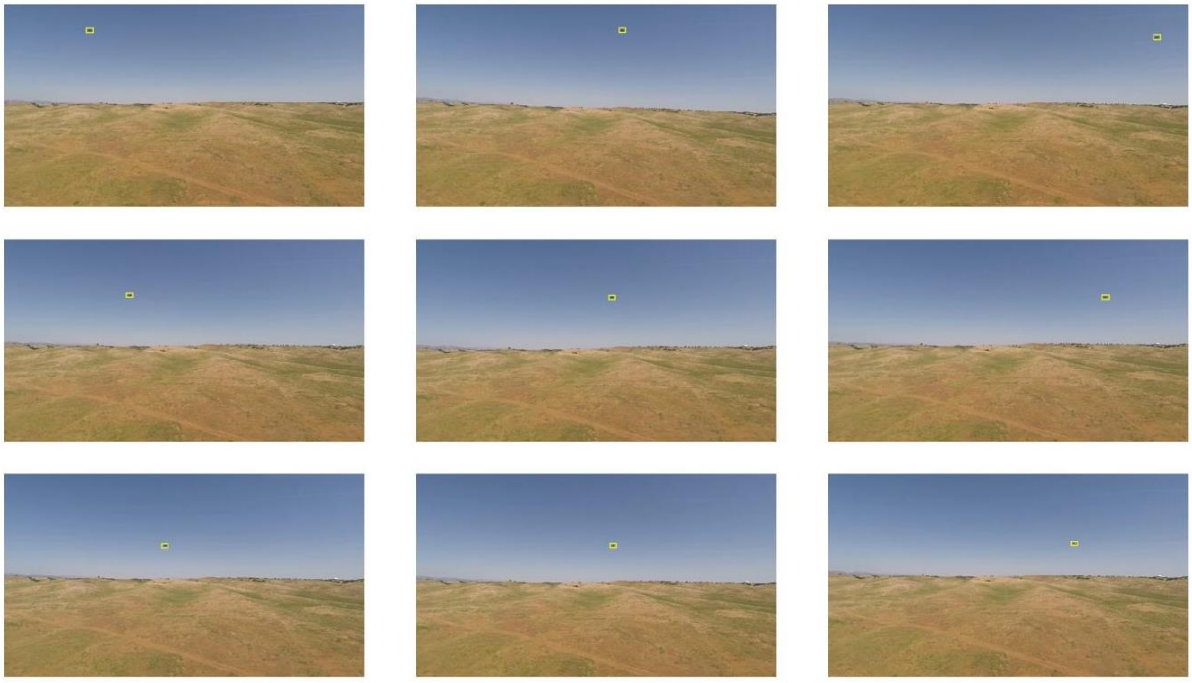


Figure 49. Montage of Waypoints at Camp Roberts
(Observed at 20 m)

Table 9. Summary of Angular Measurements at Camp Roberts
(Observed at 20 m)

Positions	Azimuth			Elevation		
	Planned azimuth angular offset (degrees)	GPS-based measurements (degrees)	CV-based estimation (degrees)	Planned elevation angular offset (degrees)	GPS-based measurements (degrees)	CV-based estimation (degrees)
A3	42.0	-	37.6	24.2	-	19.8
A2	0.0	-	8.0	24.2	-	21.1
A1	-42.0	-	-23.6	24.2	-	20.3
B3	31.0	-	24.5	16.7	-	11.1
B2	0.0	-	4.9	16.7	-	12.6
B1	-31.0	-	-15.0	16.7	-	13.5
C3	16.7	-	18.2	8.5	-	8.6
C2	0.0	-	5.1	8.5	-	8.4
C1	-16.7	-	-5.8	8.5	-	8.2

Table 10. Summary of Angular Errors at Camp Roberts (Observed at 20 m)

Positions	Azimuth (degrees)			Elevation (degrees)		
	Error between GPS and waypoint (degrees)	Error between CV estimate and waypoint (degrees)	Error between CV estimate and GPS (degrees)	Error between GPS and waypoint (degrees)	Error between CV estimate and waypoint (degrees)	Error between CV estimate and GPS (degrees)
A3	-	4.4	-	-	4.4	-
A2	-	-8.0	-	-	3.2	-
A1	-	-18.4	-	-	4.0	-
B3	-	6.5	-	-	5.6	-
B2	-	-4.9	-	-	4.1	-
B1	-	-15.9	-	-	3.2	-
C3	-	-1.5	-	-	-0.1	-
C2	-	-5.1	-	-	0.1	-
C1	-	-10.9	-	-	0.3	-

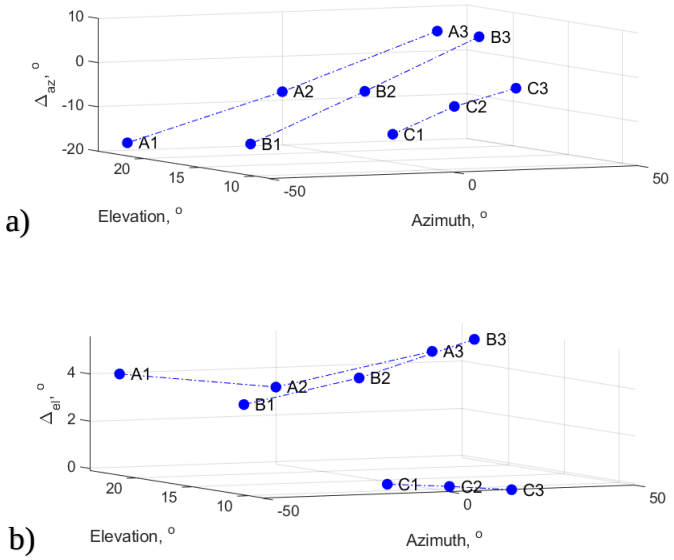


Figure 50. Visualization of Table 10 Data for (a) Azimuth Error and (b) Elevation Error

Figure 51 shows the montage of the snapshots for the nine additional waypoints flown at Camp Roberts (observed at 10 m). Table 11 summarizes the data for these waypoints, while Table 12 compares the errors. Figure 52 shows the visualization of the azimuth and elevation errors between CV estimates and planned offset in Table 12 for the different waypoints.

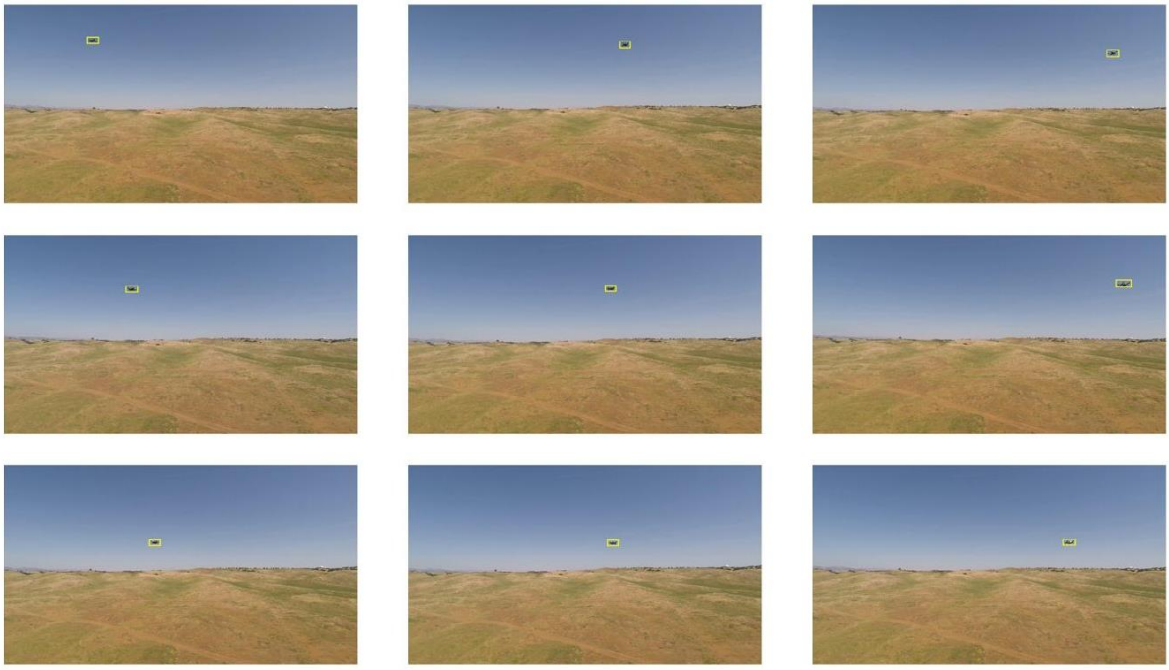


Figure 51. Montage of Waypoints at Camp Roberts (Observed at 10 m)

Table 11. Summary of Angular Measurements at Camp Roberts (Observed at 10 m)

Positions	Azimuth			Elevation		
	Planned azimuth angular offset (degrees)	GPS-based measurements (degrees)	CV-based estimation (degrees)	Planned elevation angular offset (degrees)	GPS-based measurements (degrees)	CV-based estimation (degrees)
AA3	31.0	-	27.1	16.7	-	11.8
AA2	0.0	-	10.6	16.7	-	16.8
AA1	-31.0	-	-19.5	16.7	-	15.1
BB3	21.8	-	26.2	11.3	-	10.5
BB2	0	-	7.7	11.3	-	13.5
BB1	-21.8	-	-12.5	11.3	-	11.6
CC3	11.3	-	18.7	5.7	-	5.4
CC2	0	-	8.4	5.7	-	6.3
CC1	-11.3	-	-7.4	5.7	-	6.4

Table 12. Summary of Angular Errors at Camp Roberts (Observed at 10 m).

Positions	Azimuth (degrees)			Elevation (degrees)		
	Error between GPS and waypoint (degrees)	Error between CV estimate and waypoint (degrees)	Error between CV estimate and GPS (degrees)	Error between GPS and waypoint (degrees)	Error between CV estimate and waypoint (degrees)	Error between CV estimate and GPS (degrees)
AA3	-	3.9	-	-	4.9	-
AA2	-	-10.6	-	-	-0.1	-
AA1	-	-11.4	-	-	1.6	-
BB3	-	-4.4	-	-	0.8	-
BB2	-	-7.7	-	-	-2.2	-
BB1	-	-9.3	-	-	-0.3	-
CC3	-	-7.4	-	-	0.3	-
CC2	-	-8.4	-	-	-0.6	-
CC1	-	-3.9	-	-	-0.7	-

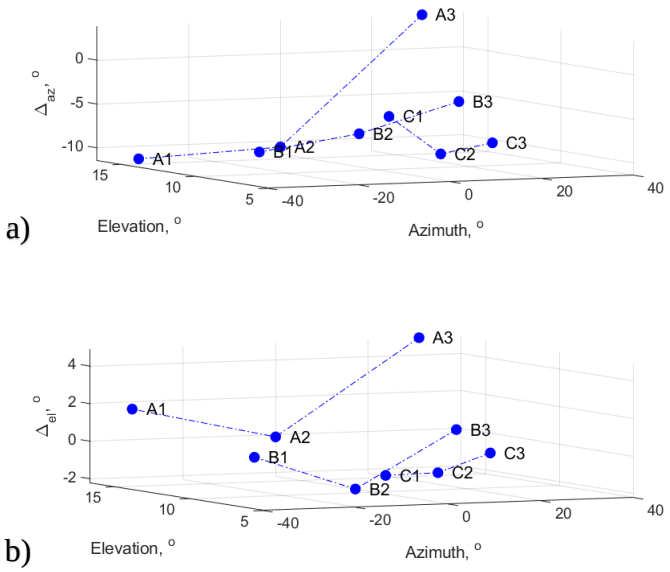


Figure 52. Visualization of Table 12 Data for (a) Azimuth Error and (b) Elevation Error

1. Results of Angular Measurements at Impossible City

In Table 7, there are no CV-based estimates for waypoints A2 and A3 as the target was not hovering within the FOV of the camera at these positions (refer to montage in Figure 47). As A2 and A3 are the outermost waypoints planned at the edge of the camera's FOV, this suggests the existence of a finite error in the relative heading and elevation angle of the camera orientation. The asymmetry in GPS-based azimuth angular measurements within each cluster of waypoints (i.e., A1–A3 cluster, B1–B3 cluster) also provides further evidence of such error. The sources of such errors could include imperfections in the drone's onboard compass (affecting heading) and accelerometer sensors as well as potential misalignment in the camera gimbal joints, collectively affecting the pitch angle of the camera.

A comparison of the GPS-based measurements and planned angular offset values shows that the target drone's accuracy in navigating to the planned waypoints is reasonable given that no angular errors (i.e., difference in angles between planned waypoints and GPS-based measurements) in azimuth and elevation exceeded 5.5^{o} in magnitude. This demonstrates the value of utilizing pre-programmed waypoints to plan and execute repeatable drone maneuvers for a variety of applications.

The CV-based angular estimation does not perform well as there are large errors of up to nearly 17^{o} and 18^{o} in magnitude when compared against planned waypoints and GPS-based measurements, respectively. These large angular errors are attributed to relative positioning errors between the target and observer drones, as well as the observer camera orientation. This is evident in Figure 47 as the montage shows asymmetry in the physical positions of the target drone within the camera FOV.

The presence of outliers in the raw data is attributed to the bounding box intermittently shifting between the actual target and spurious residual artifacts in the image frame resulting from imperfect image processing techniques in the CV algorithm. For example, foreground pixels are sometimes not completely removed before bounding boxes are constructed to isolate and localize the real target. Figure 53 shows an example of a bounding box enclosing an artifact instead of the actual target. The rapid shifts in bounding box between the target and artifacts across successive frames can be observed through visualization of the bounding boxes on the original video frames in the MATLAB software environment.

Figure 53. Bounding Box Enclosing an Artifact

2. Results of Angular Measurements at Camp Roberts

Given the lack of meaningful GPS-based measurements at Camp Roberts, comparisons in angular measurements are made against planned waypoints only. Inspecting the position of the bounding box visualization on the original video frames readily shows that all the planned waypoints have a finite offset in azimuth within the camera FOV (refer to Figure 49 and Figure 51). This is further confirmed by the asymmetry in the values of the CV-based azimuth angular measurements. This corroborates the observations of the data from Impossible City.

Given the relatively more homogeneous color tone of the landscape and sky at Camp Roberts captured within the camera's FOV compared to what was captured at Impossible City, the color-space segmentation processes in the CV algorithm resulted in much fewer residual image artifacts, which tends to draw the bounding box away from the true target position. The reduction in artifacts was observed for all 18 waypoints flown at Camp Roberts. Consequently, the position of the bounding box was significantly more stable and could steadily track the target movement over time. An example comparing the effects of residual image artifacts on the stability of the bounding box position is illustrated by Figure 54 (waypoints B1–B3 at Impossible City) and Figure 55 (waypoints B1–B3 at Camp Roberts). Clearly, Figure 54 shows the presence of significant image artifacts for videos captured at Impossible City, as indicated by numerous outliers.

As the target moves from waypoint B1 to B3 through B2, the corresponding bounding box movement is manifested as changes in the box centroid coordinates over time. The data shown in these figures are the raw values without post-processing. The sharp spikes/dips in values in Figure 54 indicate instances when the bounding boxes are formed over image artifacts instead of the true target position. This underlines the importance of minimizing such artifacts to improve the tracking accuracy and stability of the bounding box.

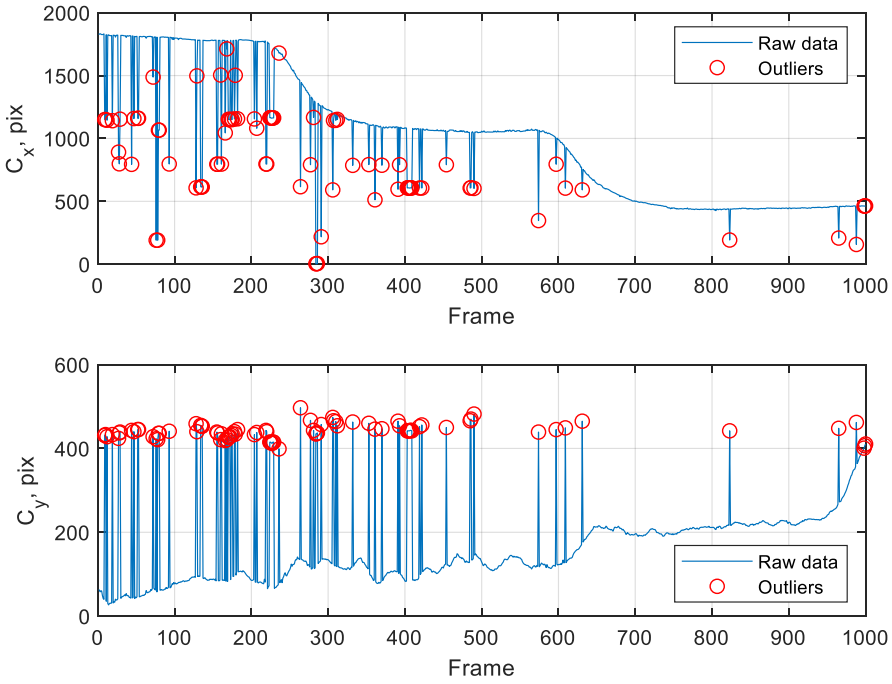


Figure 54. Variation in Centroid Coordinates (Impossible City, 20 m)

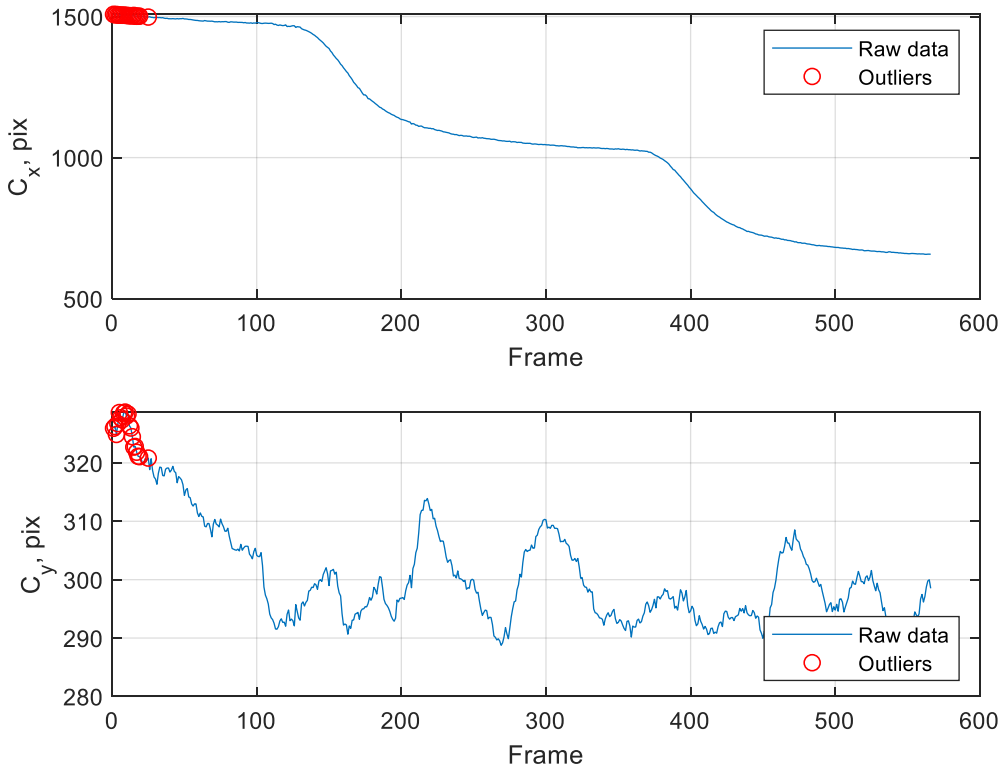


Figure 55. Variation in Centroid Coordinates (Camp Roberts, 20 m)

Given that the CV-based measurements at Camp Roberts are relatively free from the effects of image artifacts, the data from the waypoints flown at Camp Roberts is used for the evaluation of optical distortion effects and angular measurement accuracy. Figure 33 had shown that optical distortion effects are typically most pronounced at the edge of the lens. Hence, the waypoints A1-A2-A3 residing at the edge of the camera FOV would be most susceptible to the effects of such distortion. Yet, inspection of the CV-based elevation angular measurements in this waypoint cluster reveals that these values are fairly close. Ideally, the values would be the same since the target was planned to travel in a straight line across the FOV when navigating this waypoint cluster. Hence, the data does not suggest the presence of significant optical distortion effects. The assumption made is that the lens construction is symmetrical about the lens optical axis, and therefore, the observed optical properties can reasonably be assumed to be similar for symmetrical regions. The lack of observable optical distortion suggests the linear FOV mode in the camera appears to be doing its intended function, which is desirable.

For the nine waypoints observed at a range of 20 m (see data in Table 8), the azimuth errors range from 1.5^{o} to 18.4^{o} in magnitude, while the elevation errors range from 0.1^{o} to 5.6^{o} in magnitude. The preceding paragraphs already highlighted that camera orientation is a major source of error in the azimuth measurements. While sensor imperfections have already been mentioned as a contributing factor in the previous section, another factor could be the finite offset between the camera lens optical axis and the body axis of the observer drone; this is a limitation of the camera mounting bracket design used in the experiment. Hence, the physical placement of the lens could be a design consideration to further reduce azimuth measurement errors.

For the nine waypoints observed at a range of 10 m (see data in Table 10), the azimuth errors range from 3.9^{o} to 11.4^{o} in magnitude, while the elevation errors range from 0.3^{o} to 4.9^{o} in magnitude. Assuming that the elevation error arising from the pitch angle of the hovering observer drone is negligible (ideally, zero) and that the lens optical construction is reasonably symmetrical about its optical axis, the empirical errors from the CV-based elevation measurements might be a useful heuristic for design boundaries when considering CV-based angular measurements from the optical axis.

An additional observation is that the C1–C3, B1–B3, and A1–A3 waypoint clusters represent image points that are increasingly farther away from the optical center as well as the center of the image frame. The errors in angular measurements generally increase as the cluster is farther away from the optical center. This illustrates a drawback in the CV-based angular estimation technique, which assumes a constant nominal dimension for the target object. The constant nominal dimension assumption holds true only for a perfectly spherical object. As the camera observes the target at different angles, the nominal object width perceived by the camera sensor changes. For example, at an oblique angle such as waypoints at the corner of the FOV, the width of the bounding box is likely to be closer to the diagonal length of the target. This also explains why the angular errors are generally smallest when the target is observed at the waypoint cluster closest to the optical center. This effect would be diminished, as the target is farther away from the camera since the perceived change in pixel width arising from change in target orientation would be reduced.

D. IMPROVING TRACKING ACCURACY

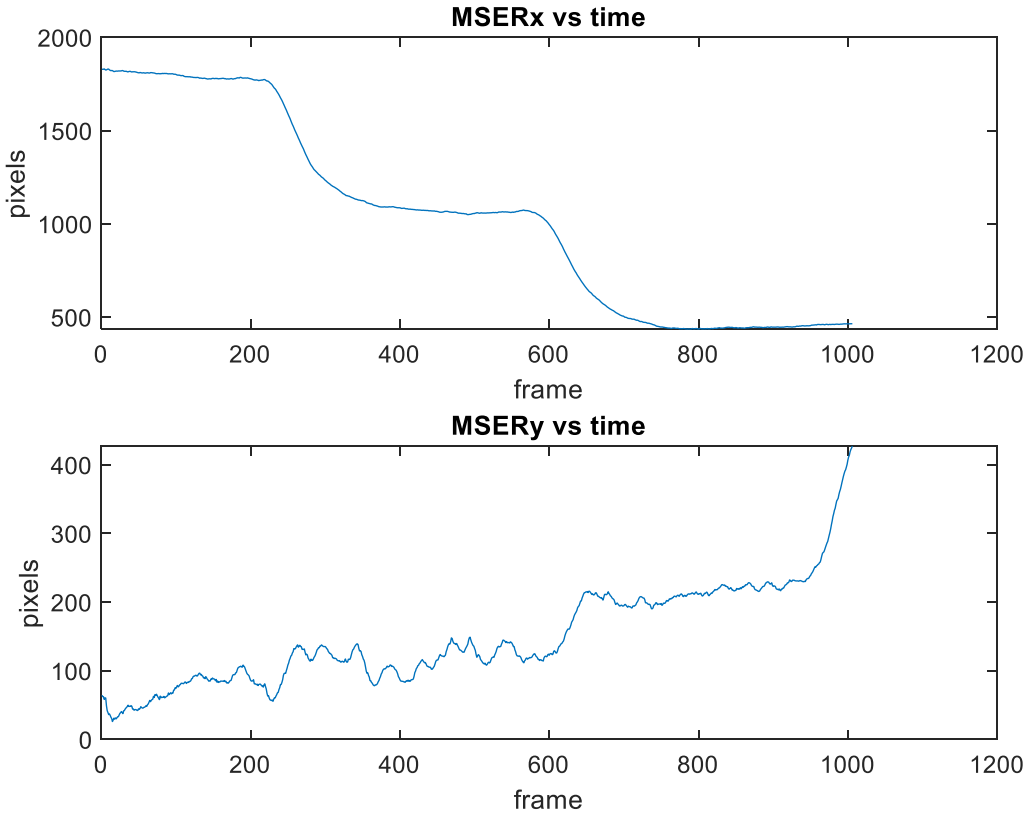
The presence of artifacts in the image frame affects the accuracy of the bounding box in tracking the true position of the target. This study has explored the use of other feature detection tools available in the MATLAB software, which are based on different feature extraction algorithms. The goal of the initial CV algorithm was to end up with an image blob representing the true target position in the frame, and use the regionprops() function to construct the bounding box around the image blob for object localization. The 2018b version of MATLAB was used to support the thesis work, and this version offers seven different types of feature detection functions. The thesis focused on three functions that were designed for blob detection, namely the detectSURFFeatures(), detectKAZEFeatures(), and detectMSERFeatures(). These functions are based on different feature detection algorithms, respectively: the Speeded Up Robust Features (SURF) (Bay et al. 2008, 346), KAZE (Alcantarilla, Bartoli, and Davison 2012, 214), and Maximally Stable Extremal Regions (MSER) (Obdrzalek et al. 2009, 107–115) algorithms.

A comparison of these feature detection functions is beyond the scope of this thesis. The focus is on the empirical results to guide the selection of the most suitable feature

detection function for the CV algorithm. The results of processing the Impossible City waypoint videos with the three types of feature detection functions show that only the detectMSERFeatures() function was able to track the target position over time. Figure 56 shows the variation of coordinates of the MSER feature point for the target navigating waypoints B1, B2, and B3 at Impossible City. The plot shows instantaneous raw values without post-processing. The difference in tracking stability is significant when compared to the raw data shown in Figure 54.

Clearly, the comparison shows that the MSER feature point is highly resistant to the effects of image artifacts and hence offers better accuracy in tracking the true target position. Visualization of the MSER feature point using a red circle overlaid on the video frames also corroborates the improved tracking accuracy as the red circle closely follows the target movement in the video. In addition, using the MSER feature point as an alternative means of target tracking is attractive as it avoids the computational penalty of having to post-process the raw signal to remove outliers. Furthermore, the post-processing in this analysis was done with the benefit of hindsight and without real-time constraint. A functional computer vision application for in-flight guidance will have to apply signal processing in real time to identify and separate noise from true target signals, and such processing will not be trivial.

Nevertheless, the main drawback of using alternative feature detection functions is the lack of other visual cues like a bounding box to indicate the size of the target object. This implies the CV-based range estimation technique cannot be employed when such alternative feature detection functions are used in isolation. A possible strategy is to employ both bounding box construction and MSER feature detection functions to improve tracking accuracy while enabling CV-based range estimation, although this incurs the penalty of additional computational cost.



MSERx refers to x-coordinate of MSER feature point; MSERy refer to the corresponding y-coordinate.

Figure 56. Variation of MSER Feature Point Coordinates for Waypoints B1–B2–B3

VII. IMPLEMENTATION

This chapter describes some initial efforts on physical implementation of a prototype system to realize the counter-UAS capability. The prototype system is meant to serve as a physical surrogate to test the computer vision and flight control algorithms. The challenges encountered in the integration of the system components and results of the flight tests for the prototype system are also discussed.

A. HARDWARE

Table 13 builds on the traceability matrix (previously shown in Figure 18) to identify the hardware used for implementing the nominal physical components. As far as possible, the hardware sourced for implementation made use of existing components in the Naval Postgraduate School laboratory inventory. A pair of telemetry radios was also used in the prototype system to allow the human operator to monitor useful information on the UAS platform such as battery level, flight mode, and error messages through the GCS software. This helps to improve situational awareness for better user control and also facilitates troubleshooting. The actual hardware setup is shown in Figure 57, while Figure 58 shows the key components on the drone.

Table 13. Actual Implementation of Physical Components.

	Physical Components	**Actual Implementation**
SYS.1.1	Touchscreen display	Windows 10 laptop with Mission Planner software
SYS.1.2	Mission planning software application	
SYS.1.3	USB port	
SYS.1.4	Device battery	
SYS.1.5	Device charging cable	
SYS.1.6	Device microprocessor	
SYS.2.1	Dual-axis joysticks	Fly Sky FS-i6S Remote Controller
SYS.2.2	Input buttons	
SYS.2.3	RC display screen	
SYS.2.4	Ground RC radio	
SYS.2.5	RC battery	
SYS.2.6	RC charging cable	
SYS.2.7	RC microprocessor	

	Physical Components	**Actual Implementation**
SYS.3.1	Flight management unit (FMU) (processor)	Pixhawk 2.1 Cube Flight Controller
SYS.3.2	GPS module	Here+ V2 Rover
SYS.3.3	Barometer	
SYS.3.4	Compass	
SYS.3.5	Gyroscope	
SYS.3.6	Airframe RC radio	Fly Sky FS-iA6B 2.4GHz 6 Channels Receiver
SYS.3.7	Rotors	DJI 2312E rotors
SYS.3.8	Vehicle battery	Turnigy 2200 mAh Li-Po battery
SYS.3.9	Vehicle battery charging cable	
SYS.3.10	FMU USB port	Pixhawk 2.1 Cube Flight Controller
SYS.3.11	Companion computer	Raspberry Pi 3 Model B board
SYS.4.1	HDMI bridge module	Auvidea B101 HDMI to CSI-2 Bridge
SYS.4.2	Camera	GoPro Hero4 Black
SYS.4.3	Camera battery	
SYS.4.4	Camera charging cable	
	Telemetry radios	mRo SiK Telemetry Radio V2 915Mhz

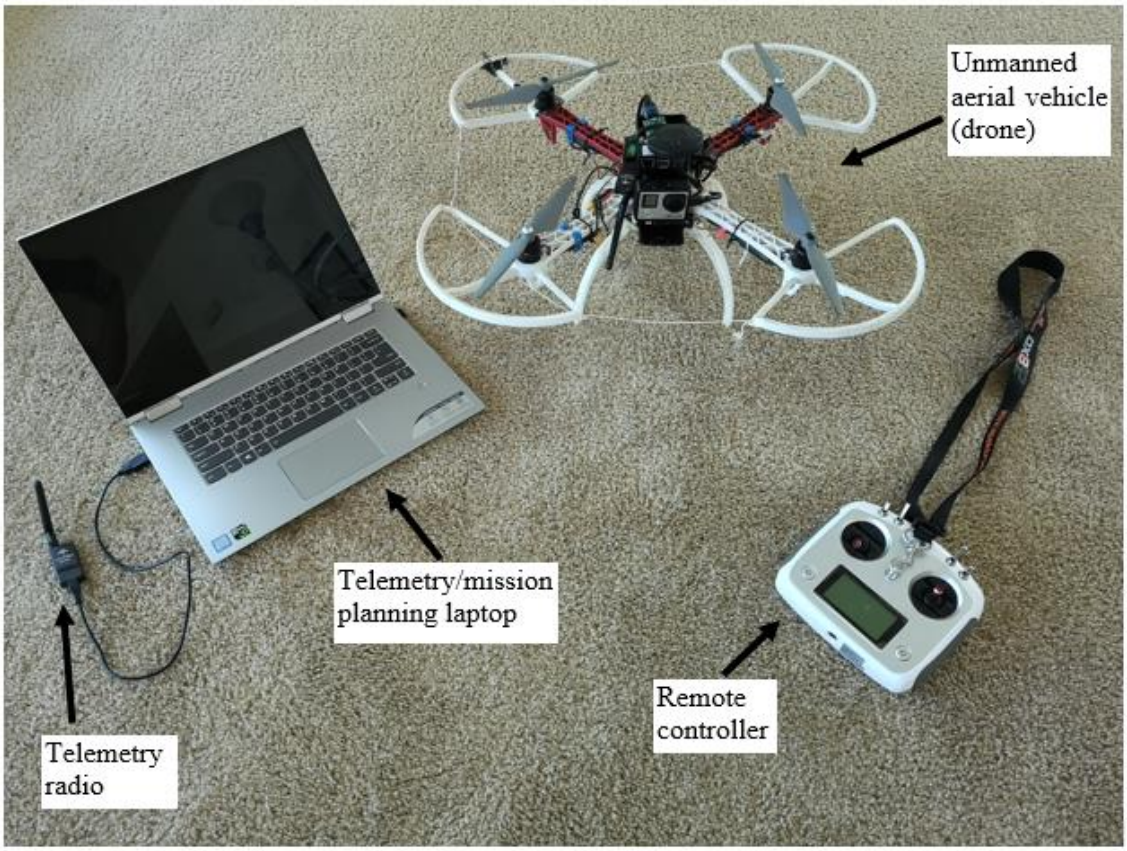


Figure 57. Hardware Implementation of Prototype System

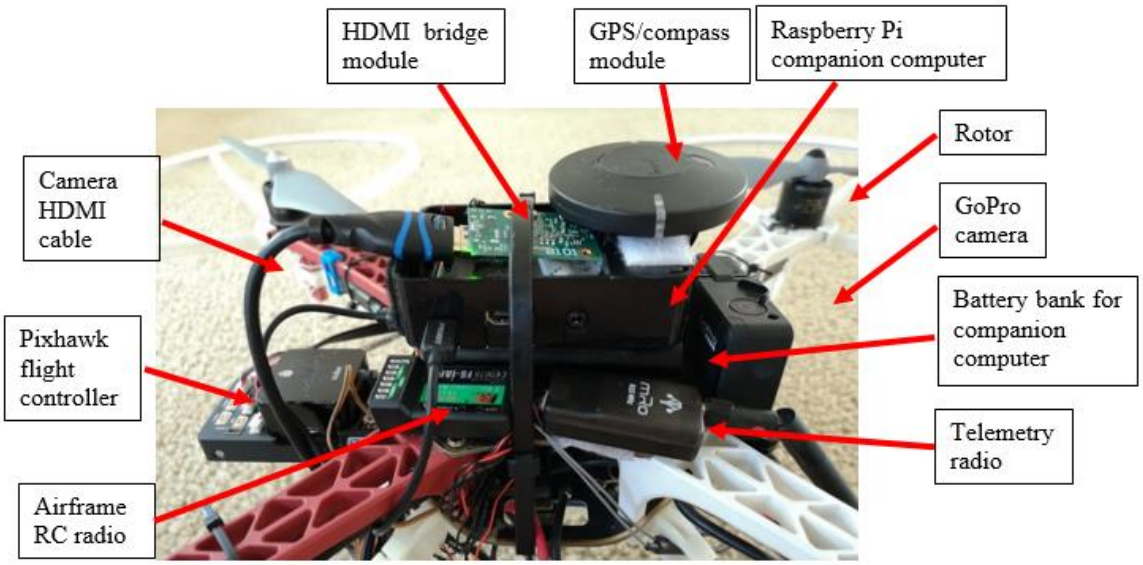


Figure 58. Key Components on the Drone

The GoPro Hero4 Black camera was used in recording the test videos for the development of the CV algorithm; hence, the same camera was selected to implement the EO sensor of the prototype system. Implementing autonomous behavior requires the computer vision and flight guidance algorithms to run on a separate companion computer as the flight management unit (FMU) is only designed to function as an autopilot, i.e., control the actuators (rotors) to achieve a desired flight command input (e.g., take-off, land, roll, pitch, yaw). In an autonomous drone, the companion computer generates the flight commands, which the FMU processes and translates into actuator controls. The Raspberry Pi 3 Model B was selected as the companion computer as it is a mature platform commonly used for electronics prototyping and enjoys a wide network of online support forums that eases troubleshooting. The Auvidea B101 HDMI to CSI-2 Bridge module was used as a hardware interface to allow the Raspberry Pi companion computer to receive video feed from the GoPro camera (Vargas 2016). This thesis does not cover details on the physical assembly and wiring connections between the hardware components as these are already well addressed by online documentation (ArduPilot Dev Team n.d.; PX4 Dev Team 2019).

B. SOFTWARE

The Mission Planner GCS software previously used for waypoint planning was used for telemetry purposes to remotely monitor the flight parameters and diagnostics status of the drone. A screenshot of the telemetry visualization in the Mission Planner software is shown in Figure 59.

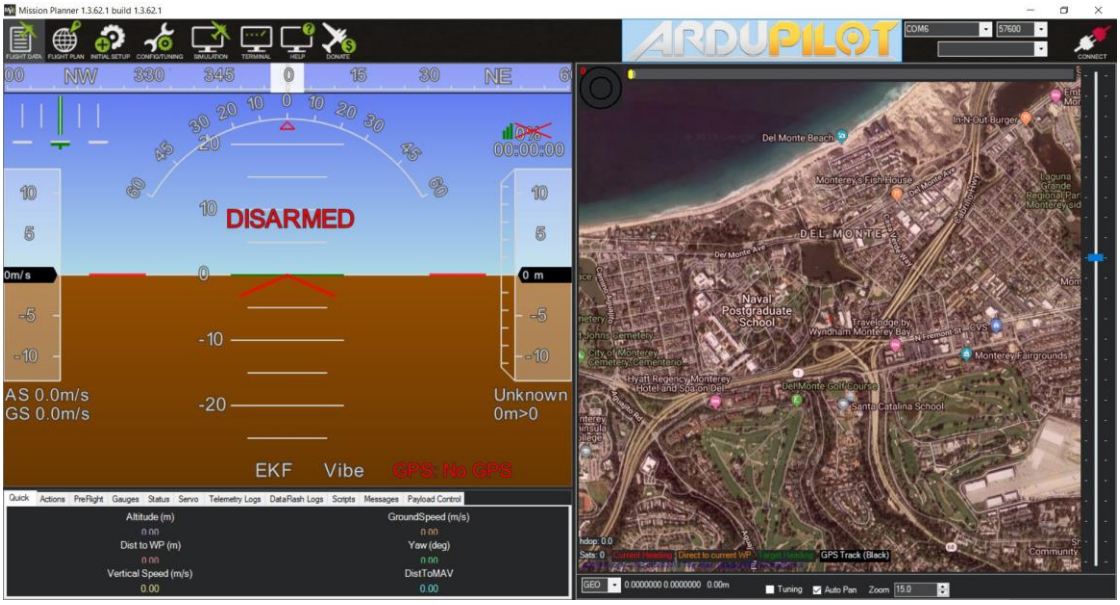


Figure 59. Screenshot of Telemetry Visualization in Mission Planner Software.

The Raspberry Pi companion computer runs on the Raspbian operating system system, which supports the Python programming language, amongst others (Raspberry Pi Foundation n.d.). Python was selected as the programming language to implement the CV and flight guidance algorithms on the prototype system as there are computer vision and drone flight control software libraries (i.e., OpenCV and DroneKit) readily available, without having to build the code from scratch. This greatly streamlines the prototyping process. As the prototype drone was planned to be tested at Camp Roberts, the MATLAB CV algorithm (refer to Appendix B) was translated into Python programming code.

User interaction with the Raspberry Pi companion computer is achieved remotely, using Virtual Network Computing (VNC). This allows the user to access and control the

desktop interface of the Raspberry Pi companion computer from another computer; in this case, the telemetry/mission planning laptop (Raspberry Pi Foundation n.d.). Hence, using VNC helps to streamline user control, monitoring, and programming of the autonomous drone system to a single computer. Establishing the VNC connection between the laptop and the companion computer requires three steps: 1) enabling the VNC Server in the Raspberry Pi, 2) installing the VNC Viewer software on the laptop, and 3) ensuring both laptop and Raspberry Pi are connected to the same wireless network. In field tests, the wireless network was provided by a smartphone in wireless hotspot tethering mode. Since the user has remote access to the desktop interface of the companion computer, such access also extends to the video output of the camera connected to the companion computer.

C. FLIGHT GUIDANCE ALGORITHM

This section describes the control objective, guidance principle, and workflow of the algorithm, as well as the software libraries used in the algorithm implementation.

1. Software Libraries

The CV algorithm presented in Chapter III is intended to help localize the target position within the FOV of the camera. Such information is further translated into guidance commands for the drone via the flight guidance algorithm. These commands are implemented through software functions in Python using DroneKit software libraries. DroneKit allows progammers to create Python programs that communicate with the unmanned vehicle via the MAVLink protocol. The software library enables the programmer to gain access to the vehicle's state and parameter information and allows direct control over the vehicle movement (3D Robotics n.d.).

2. Control Objective

As the desired capability is to intercept an intruding drone, the control objective was to stop pursuing the target when it is within an arbitrary distance of 2 m and to execute a 'return to launch' command, directing the drone would return to its launch point. The stopping distance is converted into the equivalent bounding box width using Equation (4).

3. Guidance Principle

A simple guidance principle utilizing the azimuth angular offset of the target is proposed to be tested. This is intended to test the drone's ability to pursue the target in the simpler scenario of a two-dimensional plane. Incorporating the elevation angular offset to allow the drone to pursue a target in three-dimensional space is left as future work.

At every frame of the video output from the camera, the CV algorithm would determine the azimuth angular offset of the target from the optical axis. This angular offset is then fed as the desired yaw angle for the drone to execute while constantly advancing with a forward motion vector relative to the body axis of the drone. The Pixhawk autopilot then uses the proportional-integral-navigation controller to eliminate the azimuth offset. In field tests, a forward vector of zero was used to isolate and test the yaw rotational motion. Ideally, in such a configuration, the drone should yaw in the azimuth plane to track the target as the latter moves across the camera FOV. The implementation of more complex guidance principles is left as future work.

4. Guidance Algorithm

The flight guidance algorithm workflow is described in the following steps:

1. Arm the drone and take off to pre-determined altitude.
2. Initialize the camera and start the CV algorithm to read the incoming video frame.
3. Apply the CV algorithm to detect the target and find a bounding box.
4. Compute the bounding box width and azimuth angular offset.
5. Compare the bounding box width with the control objective.
6. Intercept the target and return to launch point if the width is equal to or greater than the objective.
7. Feed the azimuth offset as the desired yaw angle input command to the drone, if the conditions in step 6 are not met.

8. Command the drone to advance with a constant forward motion vector (relative to its body axis) while executing the yaw command.

9. Go to step 2 to process the next incoming video frame and repeat all subsequent steps until the control objective is achieved.

D. INTEGRATION CHALLENGES

The integration process for the prototype system faced several challenges. Beyond physical assembly and wiring, conscious and deliberate decisions were taken to optimize the relative placement of various components on the airframe. The following paragraphs describe the challenges encountered and the mitigation actions taken.

1. Separate Power Source for Companion Computer

While the wiring diagram for the connection between the Raspberry Pi and the Pixhawk Cube FMU showed the possibility of supplying the companion computer with 5V power from the telemetry (TELEM2) port of the Pixhawk, this was not a feasible arrangement in practice as the power supply was not stable, and the Raspberry Pi was observed to repeatedly reboot itself when the Raspbian desktop environment was launched. The mitigation action taken was to add a portable battery bank to provide dedicated power supply for the companion computer. This, however, has the drawback of increasing the platform weight.

2. Propeller Guards within Camera FOV

Propeller guards were installed to protect the propellers from coming into contact with other surfaces. Nevertheless, the tips of the propeller guards appeared within the FOV of the camera and could partially obscure the target. Furthermore, under shady conditions, these tips can appear as dark areas in the FOV and be wrongly detected as false targets. Possible mitigation actions include changing the FOV of the camera to a narrower angle at the expense of reduced situational awareness. Alternatively, the mounting position of the camera can be adjusted to remove the propeller guard from the FOV, but care must be taken to balance the center of gravity of the drone vehicle. In this thesis, the FOV setting of the camera was adjusted to minimize the propeller guard obstruction.

3. Space Management and Electromagnetic Compatibility

The prototype system made use of the DJI F450 quadrotor airframe and despite the 450 mm airframe width, it was necessary to ensure all other components and wiring were clear of the four propeller blade movement areas. Consequently, there was limited space for the placement of components, which were concentrated in the center of the airframe. The prototype system made extensive use of cable ties to secure components and wiring to prevent unwanted movement. Component surfaces were attached together using velcro strips to allow easy and repeatable attachment/detachment for troubleshooting purposes.

On the other hand, the concentration of components meant that the electronics, power supplies, and antennas (for remote control, telemetry, Wi-Fi, and GPS) as well as power and data cables were all in proximity, giving rise to electromagnetic compatibility concerns. During field tests, the telemetry screen in Mission Planner showed intermittent compass calibration errors despite several attempts at compass re-calibration. It was observed that such error messages typically occurred when the autonomous drone drifted and flew in an unplanned manner. Nonetheless, the error messages could not be replicated consistently across all test flights, which frustrated attempts at troubleshooting.

The portable battery bank appeared to be a possible contributing factor as the same battery bank also resulted in intermittent magnetic interference error messages when it was mounted on the 3DR Solo drone to power the GPS module during the initial CV algorithm test flights. As no suitable alternative portable battery bank was available during the test flights, the compass error messages could not be resolved definitively. Future work, however, may need to consider the use of suitable electromagnetic shielding or appropriate separation between such components to reduce electromagnetic compatibility issues.

E. FLIGHT TEST OBSERVATIONS AT CAMP ROBERTS

Flight tests were conducted at Camp Roberts on 15–16 August 2019. Initially, when all the hardware components on the drone were assembled and tested indoors, the Raspberry Pi companion computer was able to receive video feed from the GoPro camera. These components were then disassembled for transportation to Camp Roberts. After the same components were assembled again for the field tests, the Raspberry Pi was unable to

receive video feed. Troubleshooting had isolated the fault to a possible malfunction in the HDMI bridge module. As there was no spare HDMI bridge module available, the flight tests at Camp Roberts were limited to testing arbitrary flight commands by using the Raspberry Pi to run DroneKit software function calls. The drone successfully demonstrated vertical take-off/landing, rotational (yaw), and translational motion. This gave assurance that the integration between the DroneKit software and the drone hardware was working as intended.

On the other hand, it was also observed that the yaw motion of the drone was not stable. When the drone was commanded to execute continuous 90^{o} yaw movement in a software loop (without any other motion vectors), it was not executing the yaw movements along a fixed axis; i.e., as the drone yaws, the axis of yaw rotation also drifts. Figure 60 shows snapshots of the yaw movements. A possible cause of the drift could be weak or fluctuating GPS signals given the GPS antenna is mounted in proximity to other electronics, antennas, power supply, and the plane of the rotors on the airframe, all of which could contribute to electromagnetic interference. Future designs could explore adding a GPS antenna mast on the drone as well as a GPS RTK base station to improve the drone's GPS self-localization performance.

Figure 60. Snapshots of Unstable Yaw Movements

F. FLIGHT TEST OBSERVATIONS AT IMPOSSIBLE CITY

The camera sub-system comprising a GoPro camera, HDMI bridge module, and HDMI cable was replaced by the Raspberry Pi Camera module v2 (see Figure 61). This camera module is directly compatible with the camera port of the Raspberry Pi and does not require any further modification to the prototype system. Hence, it was used as a quick solution to resolve the video feed problem. A plastic casing was also fabricated using 3D printing to physically protect and mount the camera module on the airframe (see Figure 62).

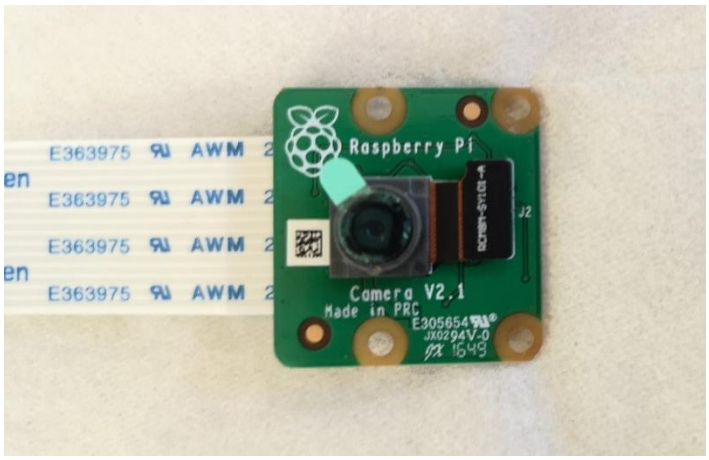


Figure 61. Raspberry Pi Camera Module v2

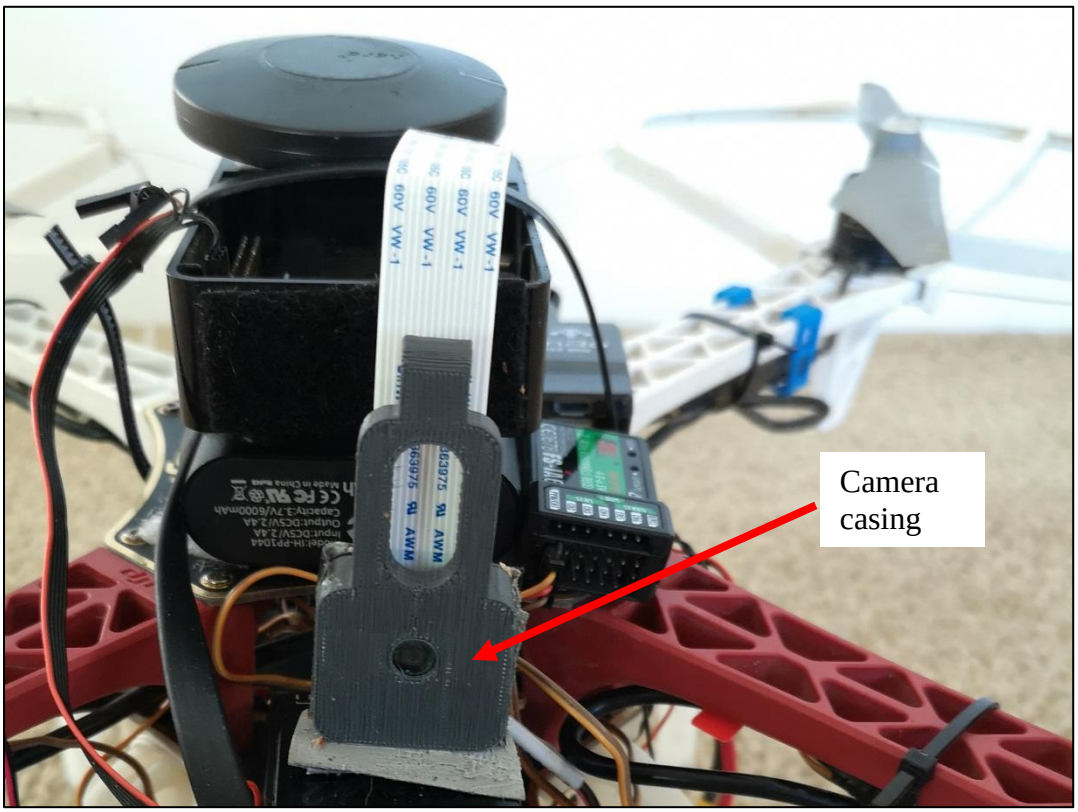


Figure 62. Camera Casing

A static target in the form of a 30 cm-wide black color cardboard mounted on a tripod was set up for the field test at Impossible City on 30 August 2019 (see Figure 63). The drone was placed 5 m away from the target board and the flight guidance algorithm

was run. Ideally, the drone was expected to approach the target board and return to launch once it achieved the control objective (i.e., reached the stopping distance). However, erratic motion was observed. When observing the video output on the telemetry laptop, it was evident that the CV algorithm was detecting erroneous targets as large patches of dark-colored vegetation in the background, which contributed image artifacts. The intermittent appearance of image artifacts generated unwanted motion vector commands that caused the drone to navigate in an erratic manner.

The flight tests also showed that the criteria for meeting the control objective need to be more robust. As the flight guidance algorithm relied solely on the bounding box width, large bounding boxes due to background artifacts triggered the drone to terminate the pursuit behavior prematurely. One possible way to make the criteria more robust in response to artifacts is to factor in additional stopping conditions such as the area of the bounding box or other distinctive bona fide target characteristics.

Figure 63. Target Board

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS AND RECOMMENDATIONS

This chapter summarizes the conclusions to the research questions posed in Chapter I and lists the recommendations for future work that would advance the efforts of this thesis in the implementation of a practical and effective counter-UAS capability.

A. SUMMARY

This thesis developed a color-space segmentation CV algorithm for object detection and localization, and presented a technique for monocular estimation of the relative range and angular position of a nominal drone target, using a COTS camera. A prototype system was also built to implement and demonstrate the concept.

Field testing was first conducted at Impossible City and Camp Roberts to record video footage of a 3DR Solo drone as the nominal drone target platform. The video recording was done with a GoPro Hero4 Black camera mounted on another hovering 3DR Solo drone acting as the observer platform. The CV algorithm was first developed based on the flight videos at Impossible City and later modified for the landscape and sky conditions at Camp Roberts.

The range estimation performance was assessed against GPS-based distances at Impossible City and waypoint-based distances at Camp Roberts. First, the upper range bound of the proposed algorithm was established to be 0.95^{o} in terms of the angular size of the target, which for the small-size UAS of width 33 cm corresponds to about 20 m range from the camera. For larger targets, such as the ScanEagle UAS featuring a 3.1 m wingspan (Huber 2018), the detection range would be in the order of approximately 190 m. Second, because of the way the bounding box is drawn after image processing (which adds pixels to the target image's edges through morphological dilation operations), the box width is always larger than the true target width. Consequently, the inverse relationship between the target width and target distance from camera means that the target distance is always underestimated. Hence, a correction factor needs to be introduced. The limited tests conducted within this study suggest that the correction factor can be as large as 0.55.

A symmetrical set of waypoints was planned to evaluate the angular estimation performance of the CV algorithm. At Impossible City, significant asymmetry in the actual waypoints flown by the target was visually observed in the video frames and confirmed by GPS measurements. This affected evaluation of the angular estimation performance at Impossible City as the asymmetry in spatial positioning of the target is confounded with true angular estimation.

The field tests were repeated at Camp Roberts with an additional set of waypoints flown at closer range to the camera. As no GPS data was available, evaluation of the angular estimation performance was compared solely against planned waypoints. For the nine waypoints observed at 20 m away from the camera, the azimuth errors ranged from 1.5^{o} to 18.4^{o} in magnitude, while the elevation errors range from 0.1^{o} to 5.6^{o} in magnitude. For waypoints observed at 10 m away from the camera, the azimuth errors range from 3.9^{o} to 11.4^{o} in magnitude, while the elevation errors range from 0.3^{o} to 4.9^{o} in magnitude. The aforementioned results of the angular estimation did not reveal any dependency in the estimation error from the actual target position within the image frame, contrary to initial expectations. Nevertheless, the results of range estimation showed that a correction factor is needed, depending on the prior camera calibration. This paves the way for future work to further investigate feasible techniques for accurate monocular range estimation. Alternatively, the target should always be positioned in the middle of the image frame (i.e., aligned with optical axis) via active gimbal control to reduce estimation errors.

The CV algorithm was not adaptive to different landscapes and sky conditions. In addition, imperfections in image processing resulted in residual artifacts in the image frame, which affected the accuracy of the bounding box in tracking the true position of the target. Flight videos recorded at Impossible City suffered from image artifacts and had to undergo post-processing to remove outliers, which would otherwise worsen the angular estimation accuracy. Such post-processing would not be feasible for real-time flight guidance. Other means of feature tracking were explored, and the MSER feature detection algorithm was found to be promising as it was observed to be resistant to the presence of image artifacts. The drawback of MSER feature tracking, however, is that it lacks the visual cue for range estimation, unlike bounding boxes with finite area, height, and width. A

possible strategy is to employ both bounding box construction and MSER feature detection functions to improve tracking accuracy.

A prototype system was built to implement the counter-UAS concept, and it successfully demonstrated vertical take-off/landing, rotational (yaw), and translational motion in field tests at Camp Roberts, based on arbitrary software flight commands. Yet, it was observed that the yaw motion of the drone vehicle was not stable. It is postulated that the unstable yaw motion might be due to weak or fluctuating GPS signals arising from the GPS antenna's proximity to other sources of electromagnetic interference on the airframe. The implementation of the prototype system encountered several integration challenges involving provision of stable power supply for the companion computer, minimizing structural obstructions within the camera FOV, space management, and electromagnetic compatibility between components on the airframe.

Field testing of the flight guidance algorithm was conducted at Impossible City. Erratic motion of the drone was observed. This is attributed to the intermittent appearance of image artifacts in the video frame, which generated unwanted motion vector commands that caused the drone to navigate in an erratic manner. The flight tests also showed that the criteria for meeting the control objective need to be more robust. One possible way to make the criteria more robust in response to artifacts is to factor in additional stopping conditions based on other distinctive bona fide target characteristics.

B. RECOMMENDATIONS FOR FUTURE WORK

Through the development and testing of the prototype system, the future work is recommended in the following areas:

- Develop an alternative CV object detection and localization algorithm to adapt to different landscapes and sky conditions. For example, deep learning-based approaches would require a training dataset of video frames with annotated drone and suitable hardware to support the computational demand, but the potential payoff is an adaptive drone detection capability.

- Improve the technique for monocular estimation of relative range and angular offset of target from the camera optical axis with lower estimation errors.

- Implement a drone vehicle with more stable aerodynamic performance. Using a stable drone platform for research helps to decouple the performance of the underlying CV and flight guidance algorithms from the inherent aerodynamic performance of the drone. This facilitates troubleshooting to improve the algorithms.

- Explore different guidance principles to optimize the performance of the drone in intercepting a target. This study implemented a simple guidance principle for the prototype system; however, further improvements in flight performance could possibly be reaped through modeling and simulation to determine the optimal trajectory control scheme and to fine tune the control gains configuration in the FMU for a practical prototype system.

The abovementioned recommendations are aimed at addressing the drawbacks and limitations encountered in the development and testing of the prototype system. These would collectively aid the implementation of a practical and effective counter-UAS capability.

APPENDIX A. MATLAB CODE FOR CV ALGORITHM (IMPOSSIBLE CITY)

This appendix contains the MATLAB code for the CV algorithm used to process all drone flight videos recorded at Impossible City.

```
clear all;
close all;
clc;

%% initialize variables

r = 5; % r is the radius of the plotting circle
j=0:.01:2*pi; %to make the plotting circle

centroids=[0;0];%initialize empty variable

%read inputs and test play the video
dronevid = VideoReader('video file.mp4');
get(dronevid);

nframes = dronevid.NumberOfFrames;

pos_2=540;%initialize tracker to centre of FOV
pos_1=960;

hbb=zeros(1,4); %create empty array for bbox to initialise
area=0;

for t= S_frame:nframes

% load the image
frame=read(dronevid,t);

hsv = rgb2hsv(frame); %convert to HSV colorspace
BW = rgb2gray(frame); %convert to grayscale

hframe=hsv(:,:,1);%get HSV frame in hue values

lightmask=hframe>=0.55; %create mask in hue frame to
eliminate foreground
lm2=BW>=130;

se = strel('disk',35);
se2 = strel('disk',5);
```

```
erode = imerode(lightmask,se); %get rid of small patches in
foreground mask
imshow(erode);

jj=and(lm2,erode);
jj2=medfilt2(medfilt2(jj));
jj3 = imdilate(jj2,se2);
jj4=imerode(jj3,se);

maskbw=immultiply(jj4,BW); % apply foreground mask to
grayscale image

imshow(frame);

regions = detectMSERFeatures(maskbw); %detect MSER region
in masked grayscale image
hold on;

if numel(regions.Location)>2
mserpoint=mean(regions.Location);
else
mserpoint=regions.Location;
end

if ~isempty(regions)
pos_1=mserpoint(1);
pos_2=mserpoint(2);

% process image for edge detection

edgeBW = edge(maskbw,'sobel');
seline5 = strel('line',5,0);
closeBW=imclose(edgeBW,seline5); %morpholoigcal close to
form image blob

m1=medfilt2(closeBW, [1 5]);
m2=medfilt2(m1, [5 1]); %median filtering to remove horizon
lines

sedisk5 = strel('disk',5);
edgemask=imdilate(m2,sedisk5); %create edge mask

edgeBWframe=immultiply(edgemask,maskbw); %apply edge mask
to masked grayscale image

sbox = regionprops(edgemask,'BoundingBox');
```

```
if ~isempty(sbox)
ctd = regionprops(edgemask,'Centroid');
a = regionprops(edgemask,'Area');
area=a.Area;

hbb = sbox.BoundingBox;

boxwidth=hbb(3);
boxheight=hbb(4);

k1=pos_1;
k2=pos_2;

centroids = ctd.Centroid;
position =[50+k1 k2; 50+k1 25+k2; 50+k1 50+k2];
value= [k1 k2 boxwidth];
RGB =
insertText(frame,position,value,'AnchorPoint','LeftBottom',
...

'FontSize',12,'BoxColor','yellow','BoxOpacity',0.5,'TextCol
or','black');
imshow(RGB);
hold on;


rectangle('Position',[hbb(1),hbb(2),hbb(3),hbb(4)],'EdgeCol
or','y','LineWidth',2);
else
k1=pos_1;
k2=pos_2;
end

end

%% define variables for time history plots

time(t)=dronevid.CurrentTime;
framenum(t)=t;
width(t)=hbb(3);
height(t)=hbb(4);
area_t(t)=area;

centx(t)=centroids(1);
centy(t)=centroids(2);
```

```
%% plot the images with the tracking
plot(r*sin(j)+ pos_2 ,r*cos(j)+ pos_1 ,'.r'); % red circle
shows the MSER point
hold off
pause(0.1)
end

%% generate time history plots after end of video

figure(2);
subplot(3,1,1);
plot(framenum,width);
title('Width vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,2);
plot(framenum,centx);
title('centx vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,3);
plot(framenum,centy);
title('centy vs time');
xlabel('frame');
ylabel('pixels');

%generate plots for 8-frame moving average
meanx=movmean(centx,8);
meany=movmean(centy,8);
meanwidth=movmean(width,8);

figure(3);
subplot(3,1,1);
plot(framenum,meanwidth);
title('Width vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,2);
plot(framenum,meanx);
title('centx vs time');
xlabel('frame');
```

```
ylabel('pixels');

subplot(3,1,3);
plot(framenum,meany);
title('centy vs time');
xlabel('frame');
ylabel('pixels');

%generate plots for 16-frame moving average
meanx2=movmean(centx,16);
meany2=movmean(centy,16);
meanwidth2=movmean(width,16);

figure(4);
subplot(3,1,1);
plot(framenum,meanwidth2);
title('Width vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,2);
plot(framenum,meanx2);
title('centx vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,3);
plot(framenum,meany2);
title('centy vs time');
xlabel('frame');
ylabel('pixels');
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MATLAB CODE FOR MODIFIED CV ALGORITHM (CAMP ROBERTS)

This appendix contains the MATLAB code for the modified CV algorithm used to process all drone flight videos recorded at Camp Roberts.

```
clear all;
close all;
clc;

%% initize variables

r = 5; % r is the radius of the plotting circle
j=0:.01:2*pi; %to make the plotting circle

centroids=[0;0];%initialize empty variable

%read inputs and test play the video
dronevid = VideoReader('video file.mp4');
get(dronevid);

nframes = dronevid.NumberOfFrames;

pos_2=540;%initialize tracker to centre of FOV
pos_1=960;

hbb=zeros(1,4); %create empty array for bbox to initialise
area=0;

for t= S_frame:nframes

% load the image
frame=read(dronevid,t);

hsv = rgb2hsv(frame); %convert to HSV colorspace
BW = rgb2gray(frame); %convert to grayscale

hframe=hsv(:,:,1);%get HSV frame in hue values

lightmask=hframe>=0.61; % create mask in hue frame to
eliminate foreground
lm2=BW<=45;

jj2=and(lm2,lightmask);
```

```
se2 = strel('disk',5); %define structuring element for
dilation

jj3 = imdilate(jj2,se2);
maskbw=immultiply(jj3,BW); % apply foreground mask to
grayscale image

imshow(frame);

regions = detectMSERFeatures(maskbw); %detect MSER region
in masked grayscale image
hold on;

if numel(regions.Location)>2
mserpoint=mean(regions.Location);
else
mserpoint=regions.Location;
end

if ~isempty(regions)
pos_1=mserpoint(1);
pos_2=mserpoint(2);

% process image for edge detection

edgeBW = edge(maskbw,'sobel');
seline5 = strel('line',5,0);
closeBW=imclose(edgeBW,seline5); %morpholoigcal close to
form image blob

m1=medfilt2(closeBW, [1 5]);
m2=medfilt2(m1, [5 1]); %median filtering to remove horizon
lines

sedisk5 = strel('disk',5);
edgemask=imdilate(m2,sedisk5); %create edge mask

edgeBWframe=immultiply(edgemask,maskbw); %apply edge mask
to masked grayscale image

sbox = regionprops(edgemask,'BoundingBox');

if ~isempty(sbox)
ctd = regionprops(edgemask,'Centroid');
a = regionprops(edgemask,'Area');
area=a.Area;
```

```
hbb = sbox.BoundingBox;

boxwidth=hbb(3);
boxheight=hbb(4);

k1=pos_1;
k2=pos_2;

centroids = ctd.Centroid;
position =[50+k1 k2; 50+k1 25+k2; 50+k1 50+k2];
value= [k1 k2 boxwidth];
RGB =
insertText(frame,position,value,'AnchorPoint','LeftBottom',
...

'FontSize',12,'BoxColor','yellow','BoxOpacity',0.5,'TextCol
or','black');
imshow(RGB);
hold on;


rectangle('Position',[hbb(1),hbb(2),hbb(3),hbb(4)],'EdgeCol
or','y','LineWidth',2);
else
k1=pos_1;
k2=pos_2;
end

end

%% define variables for time history plots

time(t)=dronevid.CurrentTime;
framenum(t)=t;
width(t)=hbb(3);
height(t)=hbb(4);
area_t(t)=area;

centx(t)=centroids(1);
centy(t)=centroids(2);


%% plot the images with the tracking
plot(r*sin(j)+ pos_2 ,r*cos(j)+ pos_1 ,'.r'); % red circle
shows the MSER point
```

```
hold off
pause(0.1)
end

%% generate time history plots after end of video

figure(2);
subplot(3,1,1);
plot(framenum,width);
title('Width vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,2);
plot(framenum,centx);
title('centx vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,3);
plot(framenum,centy);
title('centy vs time');
xlabel('frame');
ylabel('pixels');

%generate plots for 8-frame moving average
meanx=movmean(centx,8);
meany=movmean(centy,8);
meanwidth=movmean(width,8);

figure(3);
subplot(3,1,1);
plot(framenum,meanwidth);
title('Width vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,2);
plot(framenum,meanx);
title('centx vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,3);
plot(framenum,meany);
title('centy vs time');
```

```
xlabel('frame');
ylabel('pixels');

%generate plots for 16-frame moving average
meanx2=movmean(centx,16);
meany2=movmean(centy,16);
meanwidth2=movmean(width,16);

figure(4);
subplot(3,1,1);
plot(framenum,meanwidth2);
title('Width vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,2);
plot(framenum,meanx2);
title('centx vs time');
xlabel('frame');
ylabel('pixels');

subplot(3,1,3);
plot(framenum,meany2);
title('centy vs time');
xlabel('frame');
ylabel('pixels');
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

3D Robotics. n.d. “3DR Solo Drone.” Accessed April 18 10, 2019. https://3dr.com/solo-drone/.

———. n.d. “DroneKit Python.” Accessed July 10, 2019. https://github.com/dronekit/dronekit-python#overview.

———. 2015. “The Solo Controller: A Controller That’s Smarter Than Most Drones.” June 1, 2015. https://3dr.com/blog/the-solo-controller-a-controller-that-s-smarter-than-most-drones-5675479502cd/.

Alcantarilla, P.F., A. Bartoli, and A.J. Davison. 2012. “KAZE Features.” In *European Conference on Computer Vision* 7577: 214–227. https://doi.org/10.1007/978-3-642-33783-3_16.

ArduPilot Dev Team. n.d. “Choosing a Ground Station.” Accessed April 18, 2019. http://ardupilot.org/copter/docs/common-choosing-a-ground-station.html.

———. n.d. “Communicating with Raspberry Pi via MAVLink.” Accessed July 10, 2019. http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html#communicating-with-raspberry-pi-via-mavlink.

Bay, Herbert, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. 2008. “SURF: Speeded Up Robust Features.” *Computer Vision and Image Understanding* 110, no. 3 (June): 346–359. https://doi.org/10.1016/j.cviu.2007.09.014.

BBC. 2019. “Heathrow Airport Drone Investigated by Police and Military.” January 9, 2019. https://www.bbc.com/news/uk-46804425.

Coleman, David. 2014. “GoPro HERO4 Black Video and Photo Resolutions, Framerates, and Shooting Modes.” David Coleman Photography: Have Camera Will Travel. Last modified July 2019. https://havecamerawilltravel.com/gopro/gopro-hero-4-black-video-photo-modes/.

DJI. n.d. “Mavic 2.” Accessed April 18, 2019. https://www.dji.com/mavic-2/info.

EMLID. n.d. “Antenna Placement.” Accessed July 10, 2019. https://docs.emlid.com/reach/antenna-placement/.

Gettinger, Dan. 2018. “Summary of Drone Spending in the FY 2019 Defense Budget Request.” Center for the Study of the Drone at Bard College. April 9, 2018. https://dronecenter.bard.edu/files/2018/04/CSD-Drone-Spending-FY19-Web-1.pdf.

GoPro. n.d. “HERO4 Field of View (FOV) Information.” Accessed April 18, 2019. https://gopro.com/help/articles/Question_Answer/HERO4-Field-of-View-FOV-Information.

———. n.d. “What is Linear Field of View (FOV).” Accessed April 18, 2019. https://gopro.com/help/articles/Question_Answer/What-is-Linear-Field-Of-View-FOV.

Huber, Mark. 2018. “ScanEagle UAS Offers New Capabilities.” AIN Online. February 3, 2018. https://www.ainonline.com/aviation-news/defense/2018-02-03/scaneagle-uas-offers-new-capabilities.

Khoe, Wei Jun. 2018. “RSAF’s Anti-Drone Technologies in Action This Weekend in Jurong East.” *The Straits Times*, May 23, 2018. https://www.straitstimes.com/singapore/rsafs-anti-drone-technologies-in-action-this-weekend-in-jurong-east.

Liu, Lu, Feng Pan, Zhe An, and Dingyu Xue. 2017. “Target Tracking Control and Semi-Physical Simulation of Qball-X4 Quad-Rotor Unmanned Aerial Vehicle.” *International Journal of Advanced Robotic Systems* 14, no. 1 (January): 1–12. https://doi.org/10.1177/1729881416686956.

Mondragón, Iván F., Pascual Campoy, Miguel A. Olivares-Mendez, and Carol Martinez. 2011. “3D Object Following Based on Visual Information for Unmanned Aerial Vehicles.” *IX Latin American Robotics Symposium and IEEE Colombian Conference on Automatic Control, 2011 IEEE*: 1–7. https://ieeexplore-ieee-org.libproxy.nps.edu/document/6086794.

Murph, Darren. 2019. “Atlanta Still the Busiest Airport, and Beijing Just Hit 100 Million Passengers.” March 14, 2019. https://thepointsguy.com/news/atl-still-busiest-but-pek-coming/.

Obdržálek, David, Stanislav Basovník, Lukáš Mach, and Andrej Mikulík. 2009. “Detecting Scene Elements Using Maximally Stable Colour Regions.” In *Research and Education in Robotics - EUROBOT 2009* 82: 107–115. https://doi-org.libproxy.nps.edu/10.1007/978-3-642-16370-8_10.

Parekh, Himani S., Darshak G. Thakore, and Udesang K. Jaliya. 2014. “A Survey on Object Detection and Tracking Methods.” *International Journal of Innovative Research in Computer and Communication Engineering* 2, no. 2 (February): 2970–2978. https://pdfs.semanticscholar.org/25a6/c5dff9a7019475daa81cd5a7f1f2dcdb5cf1.pdf.

Parrot. n.d. “Drone Camera 4k HDR ANAFI.” Accessed April 18, 2019. https://www.parrot.com/us/drones/anafi.

Petersen, Bjorn. 2016. “Optical Anomalies and Lens Corrections Explained.” B&H. Last modified 2016. https://www.bhphotovideo.com/explora/photography/tips-and-solutions/optical-anomalies-and-lens-corrections-explained.

Pix4D. n.d. “Computing the Flight Height for a Given GSD.” Accessed April 18, 2019. https://support.pix4d.com/hc/en-us/articles/202557469-Step-1-Before-Starting-a-Project-1-Designing-the-Image-Acquisition-Plan-b-Computing-the-Flight-Height-for-a-given-GSD.

PX4 Dev Team. 2019. “Cube Wiring Quick Start.” Dronecode. Last modified September 02, 2019. https://docs.px4.io/v1.9.0/en/assembly/quick_start_cube.html#cube-wiring-quick-start.

Raspberry Pi Foundation. n.d. “Raspbian.” Accessed July 10, 2019. https://www.raspberrypi.org/downloads/raspbian/.

———. n.d. “VNC (Virtual Network Computing).” Accessed July 10, 2019. https://www.raspberrypi.org/documentation/remote-access/vnc/README.md.

Saxena, Ashutosh, Jamie Schulte, and Andrew Y. Ng. 2007. “Depth Estimation Using Monocular and Stereo Cues.” *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (January): 2197–2203. http://www.cs.cornell.edu/~asaxena/learningdepth/DepthEstimation_Saxena_IJCAI.pdf.

Shantaiya, Sanjivani, Keshri Verma, and Kamal Mehta. 2013. “A Survey on Approaches of Object Detection.” *International Journal of Computer Applications* 65, no. 18 (March): 14–20. https://pdfs.semanticscholar.org/b4a8/6f7b9d63f8bec7d9a0614b56034ccbfc4db0.pdf.

Teo, Wei Shun. 2018. “Advancing COTS UAV Capability to Provide Vision-Based SA/ISR Data.” Master’s thesis, Naval Postgraduate School. https://calhoun.nps.edu/handle/10945/60353.

Vargas, Aldo. 2016. “Computer Vision Using GoPro and Raspberry Pi.” *Altax* (blog), August 17, 2016. https://altax.net/blog/computer-vision-using-gopro-raspberry-pi/.

Yuneec. n.d. “Yuneec Mantis Q.” Accessed April 18, 2019. https://us.yuneec.com/mantis-q.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California