3-23-2018

# Stabilized RPA Flight in Building Proximity Operations

Michael M. Kaniut

# STABILIZED RPA FLIGHT IN BUILDING PROXIMITY OPERATIONS

THESIS

Michael M. Kaniut, Captain, USAF

AFIT-ENV-MS-18-M-212

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

**AIR FORCE INSTITUTE OF TECHNOLOGY**

**Wright-Patterson Air Force Base, Ohio**

AFIT-ENV-MS-18-M-212

# STABILIZED RPA FLIGHT IN BUILDING PROXIMITY OPERATIONS

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Systems Engineering

Michael M. Kaniut, BS

Captain, USAF

March 2018

AFIT-ENV-MS-18-M-212

# STABILIZED RPA FLIGHT IN BUILDING PROXIMITY OPERATIONS

Michael M. Kaniut, BS

Captain, USAF

AFIT-ENV-MS-18-M-212

## Abstract

The thesis seeks a solution to the requirement for a highly reliable and capable Unmanned Air Vehicle (UAV) to support a wide array of missions and applications that require close proximity flight to structures. The scope of the project includes the drafting of a concept of operations (CONOPs) describing how the mission requirements might be met using the sensor, operators, and air vehicle described here in. The demonstration of the wall-following section of that CONOPs is performed by cart testing a custom algorithm and evaluating its ability to react to its environment. Finally, a flight test was performed to characterize the capabilities of an RTK-GPS system to stably hold a UAV in a single position, and minimize vehicle yaw, as a potential means of minimizing environmental sensing requirements in GPS permissive environments. The results for RTK-GPS were, position hold standard of deviation 8.0 x 10.1cm at a 5m flight altitude, and 17cm x 12.7cm at 8m flight altitude. Yaw variation results were a standard of deviation of 1.7° at 5m and 3.7° at 8m. The LIDAR wall-following tests proved the feasibility of using a decision tree style coding approach to proximity flight near a structure, but still has some changes that should be considered before being used operationally.

## Acknowledgments

I would like to express my sincere appreciation and gratitude to Dr. Jacques for his support, guidance, and encouragement throughout the creation of this thesis. This topic held a lot of rabbit holes which would have been easy to turn into parapets, thanks for keeping me on an achievable track. I would also like to acknowledge Jeremy Gray for his patience and tutelage in the use and application of python these past several months. I began with little introduction to scripted code writing or structure. I can now proudly claim myself a python novice+ with a newly instilled interest in software coding. Finally, I would like to thank my wife for her patience with my long nights, stressed demeanor, and likely receding hairline these past several months, I could not have done this without your support and love.

Michael M. Kaniut, Capt, USAF

# Table of Contents

Page

Abstract ........ iv

Acknowledgments ........ v

List of Figures ........ ix

List of Tables ........ xi

I. Introduction ........ 1

1.1 General Issue ........ 1

1.2 Problem Statement ........ 2

1.3 Research Objectives ........ 3

1.4 Research Focus ........ 4

1.5 Investigative Questions ........ 5

1.6 Methodology ........ 5

1.7 Assumptions/Limitations ........ 7

1.8 Materials and Equipment ........ 8

1.9 Implications ........ 9

1.10 Preview ........ 10

II. Literature Review ........ 11

2.1 Introduction ........ 11

2.2 UAV Flight Near Structures ........ 11

2.3 Sensors Review ........ 19

2.4 Summary ........ 21

III. Architecture ........ 22

3.1 Introduction ........ 22

3.2 Overview ....................................................................................22

3.3 Architecture ....................................................................................23

3.4 Wall Following ....................................................................................29

3.5 Chapter Summary ....................................................................................39

IV. RTK GPS Results and Discussion ....................................................................................40

4.1 Chapter Overview ....................................................................................40

4.2 Results ....................................................................................40

4.3 Discussion ....................................................................................46

4.4 Summary ....................................................................................48

V. Wall Following Results and Discussion ....................................................................................49

5.1 Chapter Overview ....................................................................................49

5.2 Results ....................................................................................49

5.2.1 Distance Adjustment Tests ....................................................................................50

5.2.2 Corner Negotiating Tests ....................................................................................52

5.2.3 Impinging Walls Test ....................................................................................56

5.3 Discussion ....................................................................................58

5.4 Summary ....................................................................................59

Conclusions and Recommendations ....................................................................................60

6.1 Overview ....................................................................................60

6.2 Conclusions of Research ....................................................................................60

6.3 Significance of Research ....................................................................................63

6.4 Recommendations for Action ....................................................................................63

6.5 Recommendations for Future Research ....................................................................................65

6.6 Summary ....................................................................................................66

Appendix A. Concept of Operations (CONOPS) ..........................................................67

Appendix B. Wall Following Code ..............................................................................71

Appendix C. Logic Matrix............................................................................................80

Bibliography ...................................................................................................................83

## List of Figures

Page

Figure 1. Path Planning Approaches [22] ........ 16

Figure 2. Operational Activity Tree ........ 25

Figure 3. Full System Operational Activity Diagram ........ 26

Figure 4. Wall Following Activity Diagram ........ 28

Figure 5. LIDAR Regions of Interest ........ 30

Figure 6. LIDAR Output and ROI Segregation ........ 31

Figure 7. Data Gating Example ........ 32

Figure 8. Decision Tree Used for Logic Code ........ 33

Figure 9. Table Top Test Example ........ 35

Figure 10. RTK GPS Test Set-up ........ 37

Figure 11. Pixhawk 2 Autopilot ........ 38

Figure 12. BBB and LIDAR ........ 39

Figure 13. Location Measurement Example ........ 41

Figure 14. Test 2 Real (L) vs Onboard (R) Location Measurement ........ 42

Figure 15. Test 1 Real (L) vs On-board (R) Location Measurement ........ 42

Figure 16. Yaw Angle Measurement Example ........ 43

Figure 17. Measured (Top) vs Onboard (Bottom) Yaw Variance Test 1 ........ 45

Figure 18. Measured (Top) vs Onboard (Bottom) Yaw Variance Test 2 ........ 46

Figure 20. Algorithm Output ........ 50

Figure 21. “Too Far” Test Set-up and LIDAR ................................................................ 51

Figure 22. “Too Near” Test Set-Up ..................................................................................... 51

Figure 23. “Too Near” and “Too Far” Example Output.................................................. 52

Figure 24. Open Corner Test Location ............................................................................. 53

Figure 25. Open Corner Returns and Command Example .............................................. 54

Figure 26. Closed Corner Test Set-up................................................................................ 55

Figure 27. Closed Corner LIDAR Vision .......................................................................... 55

Figure 28. Approaching Corner ......................................................................................... 56

Figure 29. Impinging Walls Set-up.................................................................................... 57

Figure 30. Impinging Walls LIDAR Returns ................................................................... 57

Figure 31. Closing Corridor............................................................................................... 58

## List of Tables

Page

Table 1. Logic Table Portion ........................................ 33

Table 2. Sensor Parameter Limits ........................................ 47

# STABILIZED RPA FLIGHT IN BUILDING PROXIMITY OPERATIONS

# I. Introduction

## 1.1 General Issue

This thesis determines the current potential for a small unmanned air vehicle (UAV) to perform proximity operations beside structures precisely and safely. A vehicle capable of charting its own path to an objective frees an operator from having to intervene while performing their primary mission objectives. In addition to this, the vehicle must be able to accomplish its course corrections and path finding with minimal operator interaction, in order to minimize the total bandwidth dedicated to simple vehicle operations. The majority of the bandwidth allocation is intended to be used for the mission sensors downlink to the operators.

This topic is highly relevant to enterprise wide DoD goals, as the use of UAVs to reduce workload and expand situational awareness is increasingly relevant. Additionally, small tactical UAVs are just beginning to make in-roads into warfighter repertoires[1]. Tactical UAVs offer impressive situational awareness (SA) improvements and agile weapons platforms but are limited by having to dedicate soldiers to their operation, and thereby removing soldiers from active engagement with hostile forces. Further, a soldier actively operating a UAV is less likely to be able to maintain awareness of their own tactical situation, potentially putting them in danger of being compromised by rapidly changing battlefield conditions. Because of all of these concerns, the DoD has been actively and aggressively funding research in, and deployment of autonomous systems. Most famously the FCS[2] program of the early part of this century sought to inject autonomy into dozens of land vehicles and weapon systems. While that program proved overly ambitious for the technology of the time, the current technological art of the possible has begun to bring many of those projected technologies into reality.

This specific research project has been sponsored in order to demonstrate a capability for sensor packages requiring proximity flight to structures. This research then fits into the larger DoD autonomy research strategy of reducing the cognitive workload on operators, and extending the capability of operators to comprehend their tactical environment and situation [3]. The intent, therefore, is to demonstrate how the application of simple algorithms in vehicle navigation might significantly unload the SA requirements for an operator and allow a far more persistent and attentive focus on mission sensor data, and further support active warfighting actions.

## 1.2 Problem Statement

The difficulty of proximity building operations is tied to the complexity of any given structure face, features, environmental variation, incomplete and inaccurate knowledge of the building[4], and inaccuracy of navigation aids around structures[5]. These situational awareness concerns are not ideal for a human operator[6]. Such problems will frequently result in task overload, mission failure, or mishaps, as too much of the operator's attention is required for the task of keeping the vehicle in a safe state. Ideally a machine capable of maintaining some basic level of situational awareness, and having the means to make changes in its attitude and flight profile given those influences would be a marked improvement. Building on that, the ability to build a map and localize to that map would allow the vehicle to react quickly and decisively to any changes in its surroundings or flight dynamics. The end goal being to reduce or eventually eliminate the overall situational awareness burden on the human operator while performing their primary sensor mission.

Compounding the issues related to this solution are the problems inherent to requirements of aircraft, in that any solution must be lightweight, use minimal power, and have a small enough form factor to be carried on the airframe without demanding too large a portion of its power so that endurance is not excessively undermined. These restrictions tend to push solutions to off-board computing with high bandwidth radio communication links to a ground station. The result is an inherent weakness to jamming, additional latency in system responses to stimuli, and a potential for spoofing or other electronic countermeasures.

## 1.3 Research Objectives

The objective of this thesis is to build a candidate software and hardware architecture, plus a concept of operations (CONOPS), to support autonomous proximity operations of a multi-rotor UAV. This can be accomplished utilizing some combination of sensors, and autopilot algorithm, such that navigation is accomplished by supplementing GPS inputs. The intention being to allow highly stable and consistent operations within varied/multiple structure environments. This will be accomplished in a controlled environment where building features, or simulations thereof, can be closely introduced in attempts to measure the performance of the designed architecture and determine potential edge case concerns for the algorithms. Finally, the use of Real-Time

Kinematic (RTK) GPS will be evaluated as a means of providing the autopilot with additional information for locating and navigating near structures.

## 1.4 Research Focus

The focus of this research will be on the creation of a suitable small UAV architecture for proximity structure flight that supplements GPS with sensor inputs for guidance [10] and utilizing commercial off the shelf (COTS) componentry and computer visualization methods [11]. The ability of the system to identify proximity features and develop appropriate reactions to those inputs that are encroaching on its operational path will be tested.  Ideally the system will be able to intake these data points and adjust a flight profile actively to allow stable flight along a structure wall. This capability should be robust enough to take on the task of complex vehicle flight path adjustment and creation, in order to allow a human sensor operator to focus their attention on the mission sensor data.

Additionally, the ability of an autopilot to hold a position through the use of RTK GPS will be tested. This system is generally considered the highest accuracy possible for a GPS system on a UAV, and thus will inform how many, and what sort of on-board environment sensing is necessary for controlled near structure flight. Complications and dangers associated with this method will also be explored.

By intelligently minimizing the data inputs needed for effective situational awareness, and thereby limiting the processing requirements for the on-board system, an architecture capable of keeping the overall system within specified tolerances of nearby structures was designed.

## 1.5 Investigative Questions

IQ1. What sensors might be used to accurately navigate a UAV through complex urban environments along a structure?

IQ2. How can LIDAR be used to effectively maintain the stability and position of a UAV along a building face?

IQ3. How precisely can a multi-copter be flown in close proximity to a building?

IQ4. How precisely can RTK-GPS hold a multi-copter stationary?

IQ5. What is the yaw variance of a sensor in hover?

IQ6. What is the yaw variance of a sensor in motion?

## 1.6 Methodology

The methodology of this thesis follows this sequence.

A CONOPs (Appendix A) for the thesis experimentation effort was built to both verify the project intent against customer expectations, and to begin the architecture design process. This CONOPs was evolved with the customer to ensure that all parties were on the same page regarding use and deployment realities. Using this CONOPs also cemented the operational activities, system actors, system boundaries, and capabilities needed for the system, which were used to inform all other portions of the effort going forward.

Second, the architecture was designed to meet the operational activities discussed in the CONOPs. This began with the top-level systems definition process; all operational activities were matched to associated system requirements, capabilities, and informed the basic design attributes of the UAV and associated systems, in a classic system engineering development cycle.

Third, an algorithm for mapping UAV location with respect to nearby structures, maintaining flight stability, vehicle direction, and consistent speed was written to support the architecture as defined in the 2nd task. The algorithm used the inputs from the designed sensor suite to update its flight parameters continuously and thereby remain within the sensors operating parameters.

Fourth, these efforts were combined in a prototype vehicle to test a portion of the CONOPs, a portion deemed most technically valuable for the effort overall, and directly relevant for follow-on research. The testing proved the legitimacy of the overall CONOPs and designed architecture and gave the customer confidence that the research being conducted for them is creating a useful final outcome. Shortfalls found during this stage allow follow-on efforts to avoid the same issues and support continued technical progress.

The explicit logic functions used in this thesis will be more thoroughly detailed in Chapter IV, but the basic system functions take the outputs of the Light Detection and Ranging (LIDAR) degree and distance measurements and convert them into three regions and various stand-off distances that define the system reactions. The system assumes a right facing mission sensor system. The three regions are; forward, right, and left. These are defined by a varying width Region of Interest (ROI) measurements centered perpendicular to the face of the front, right and left of the vehicle. The front ROI expands and reduces based on the operator defined optimal forward speed on the vehicle. The right ROI holds an “Ideal distance” band defined by the operating parameters of the mission sensor. Finally, the left ROI is used to identify potential obstacles impinging on the vehicles path of travel from nearby structures, trees, etc. Combining these three ROIs give sufficient (SA) to build near comprehensive (albeit reactionary) vehicle control logic.

In addition to these sensor-based approaches, the use of RTK-GPS to localize the vehicle will be investigated. This high precision GPS solution might be capable of providing a sufficiently accurate location for vehicle flight near buildings, or at least static station keeping. The positional variance in all directions will be reviewed to determine initial sufficiency for this CONOPS.

Finally, all outputs from this thesis are relevant to follow-on efforts in on board sensor-based navigation research. Downsides or upsides to sensor choices, arrangements, and control algorithms have been cataloged to support future research in this area, and allow students and researchers to focus on the central concepts of the thesis, not the peripheral design questions.

## 1.7 Assumptions/Limitations

It is assumed that the mission sensor technology will be eventually developed to a maturity and size where a multi-copter UAV will be a viable platform for its deployment. It is assumed that the operator of the UAV retains LOS communications with the UAV. It is assumed that only basic information is known about the targeted structure, such as location, height, and other data that can be reliably determined by inspection from the ground station with prior overhead maps.

This specific design, it should be noted, is not the only possible system that might be capable of meeting the requirements of the CONOPs; in fact, variations in surface type (glass vs brick), weather conditions (fog/smoke), etc, might prove to significantly degrade the performance of this specific design iteration. This thesis was completed with the expectation of clear atmospheric conditions, and solid wall surfaces to expedite evaluation of the architecture. It may very well be that sonar, visual odometry (VO), or some combination of sensor suites will be more inherently resilient in varied environmental conditions, especially as advances are made in those two sensor technologies.

This thesis will not attempt to demonstrate the capabilities of the mission sensor suite on a UAV. This thesis is not going to address autonomous navigation around a building as operators will be in the loop for mode changing and supplementary awareness. Only stable navigation and position holding will be demonstrated in this thesis, feature tracking and other possible capabilities should be accomplished under follow-on efforts.

## 1.8 Materials and Equipment

This thesis was accomplished using a hex-copter UAV (6 individual arms, motors, and rotors), which allows significant attitude control and redundant lift in case of motor failures, and capacity for payloads. The UAV was equipped with a Pixhawk2 autopilot device which is embedded with all necessary logic for controlling a hex-rotor vehicle through all stages of flight, and even some simple flight profile following logic. The Pix2 includes an onboard IMU for attitude measurements. A Here+ GPS/Compass antenna was joined with the Pix2, which also allowed exploring the use of Real-Time Kinematic GPS. A URG-04LX LIDAR system was chosen for its low power usage, light weight, scanning range, and simplicity of interface with the companion LINUX computer. The LINUX companion computer is a BeagleBone Black system, capable of ingesting the LIDAR data and running it through the designed flight logic script, written in Python. That script outputs a control message which is bundled in MAVlink, which a Pix2 is able to directly ingest, and translate into vehicle body frame velocity commands.

## 1.9 Implications

Successfully building this architecture and demonstrating its efficacy will partially meet the requirements of the sponsor organization for their desired sensor platform. Beyond the immediate scope of this thesis, a successfully demonstrated computer navigation suite based on proximity operations would be highly valuable to a wide variety of both public and private organizations in construction, building maintenance, urban military operations, and many more.  A demonstrated flight path correcting algorithm also provides a potential breakthrough for urban warfare tactics to include small UAVs to support ground forces. Such a breakthrough has the potential to enormously increase situational awareness for squad level operators in the highly complex urban warfighting environment, or even establish unique and previously impossible firing positions based on mobile UAV weapons platforms. This thesis will provide a solid foundation for such a system, and enable more in-depth research in UAV navigation and autonomy.

## 1.10 Preview

In the next chapter, an extensive literature review was completed to understand the current state of proximity flight research and to explore available sensors, and wall following algorithms. Chapter III outlines the thesis methodology. Chapter IV the results of the RTK station holding tests. Chapter V details the results of the wall following algorithm.  Finally, Chapter VI discusses the conclusions of this research and recommends future work, including potential ways to broaden the capabilities of the algorithms used in this thesis.

# II. Literature Review

## 2.1 Introduction

This chapter, will explore the various existing research that has been done within the field of UAV based flight alongside structures, both indoors and outdoors. A review of the various approaches to environment sensing and obstacle avoidance systems implemented on robotic systems will be detailed. A sampling of the algorithms involved in wall following, localization and mapping, and path planning will be evaluated and presented. Finally, across these various disciplines, the most common sensors and sensor configurations will be detailed and evaluated for application to the CONOPs of this thesis.

## 2.2 UAV Flight Near Structures

The task of robotic flight and maneuvering around environments has been researched extensively in the past 20 years. Research reaching back to the early 90s, such as [7] and [8] , discuss the topic of robotic environment sensors and wall following. During this era, multi-rotor aircraft had not yet begun to be built, due to as yet limited existence of cheap flight controllers and high enough battery energy density to make the multi-rotor configuration feasible. In the following 15 years the multi-rotor benefitted enormously from the research and investments made in the miniaturization and accuracy of small sensor systems and processing boards, making this architecture feasible.

As technology matured to enable the expansion of computer vision techniques, and small scale efficient processing, much work was done in the area of obstacle avoidance and simple wall following codes. These were largely applied to ground based vehicles [9], but this research will provide the backbone to the work being done today in computer vision on flying platforms.

### 2.2.1 Optical Flow Obstacle Avoidance

In the discipline of robotics and obstacle avoidance, one of the most common and deeply researched methods is called “optical flow”. Optical flow is an algorithm developed by analyzing the translation of points/features between frames of a visual sensor. The resulting “vector map” can be used to evaluate whether there are objects in the vehicles vision which are on an interference trajectory as demonstrated in [4] , additionally these can be used to augment the IMU for motion sensing.

This approach to obstacle avoidance has some pros and cons. The most immediate positive for this approach is that the depth of consideration is as good or bad as the sensor selected. As demonstrated in [10], this allows a UAV to operate in complex multi-structure environments with many, varied distances, and potential obstacles to track. The potential concerns are related to the way that optical flow generates its vector maps. Optical flow is heavily dependent on being able to identify features to track between frames, in situations where the algorithm fails to find usable features or track a pattern and becomes confused by a change in lighting, a repetition of the same feature, or non-static points, and the algorithm quickly loses its efficacy.

The optical flow approach has a great deal of impressive results available in a wide variety of environments, but as the complexity of the lighting, features of the obstacles, and static nature of the scene change the algorithms become correspondingly more difficult and error prone [10]. These concerns limit the immediate opportunity for use in the current research, which will be expected to be used in unknown lighting conditions, with no guarantee of static surroundings, or non-repetitive features.

### 2.2.2 LIDAR Obstacle Avoidance

The use of LIDAR for obstacle avoidance has a very well-established precedence. In usage with ground vehicles, there are many examples in the modern application of LIDAR for the growing autonomous vehicle research sector. For example, in [11], LIDAR is used for environment recognition and lane keeping. In [12] LIDAR is used for localization in an urban environment. Both of these examples show the broad and expanding role of LIDAR in autonomy generally. In addition to this, the investments made into the technology have significantly reduced the size, price, and power requirements making feasible their expansion into other markets.

This rapid development has since bled into UAV applications. For example, in [13] the use of LIDAR for a LOWAS (LIDAR Obstacle Warning and Avoidance System) is demonstrated on a UAV performing low level flight in an urban environment. This application shows the efficacy of increasing the role of LIDAR in UAV applications and autonomy. Where a platform has sufficient power to integrate LIDAR into the autonomy suite, there is an ever-growing library of research into control algorithms and management for navigation to utilize.

### 2.2.3 Wall Following Algorithms

The creation of wall-following robots has a very long and well researched background. In 1992, [8] presented an application of ultrasonic sensors to inform a wall following algorithm on a ground vehicle. From there, significant advances have been made. There followed examples of using genetic algorithms to adjust and optimize a wall following code in [14], corridor navigation with sonar in [15], and eventually applications for UAVs such as in [16].

The basic structure and efficient coding required for a wall following algorithm make them ideal for light processing applications that require rapid outputs and continuous operation [16]. They can utilize a wide variety of ranging sensors, and as simple algorithms, are good candidates for optimization techniques. The downsides to these algorithms are they are only as effective as the sensors informing them, and they are inherently reactionary. In order to explore a structure thoroughly without requiring direct flight control by the operator, additional path finding and localization methods are necessary.

### 2.2.4 Localization and Mapping

The research that falls under "localization and mapping" has a number of different facets. The most widely published and investigated approach is called Simultaneous Localization and Mapping (SLAM). This approach takes the problem of navigating an unknown/ un-surveyed environment by, as suggested in the name, building a model of its environment and then localizing itself within that "map". This allows a level of spatial awareness that can be utilized to navigate otherwise highly complex terrain features [9], or indoor environments [17].

SLAM can be accomplished with any number of sensors, but most researchers use one of two different designs. The first is LIDAR based, which operates by building a continuous point cloud of the area around the vehicle, while tracking its movements with internal IMU sensors and pose tracking algorithms like Extended Kalman Filters (EKF) [18]. These maps can be transmitted off-board for operator situational awareness or high precision interior surveying, depending on application. The second is visual odometry (VO) and visual SLAM, and they require some number of cameras, ranging from one (monocular VO) [19] to many [20]. These use the same pose tracking with the IMUs as other methods, in conjunction with point tracking in the images. Where LIDAR point maps will generally only be capable of tracking several explicit features, VO has the capacity to track hundreds of individual features across frames. This provides an opportunity to more accurately define the vehicles location, but comes at the cost of processing speed, which drops significantly with the tracking of so many individual features [21].

### 2.2.5 Path Planning Algorithms

Path planning is a complex algorithmic operation which has been researched and implemented in dozens of different autonomous and logistical solutions. The goal of any path planning algorithm is to resolve a large solution space into an optimal (or near optimal) solution that meets the constraints of the algorithm [22]. The algorithms utilized to accomplish this vary in approach but fall into five main categories, as shown in Figure 1.

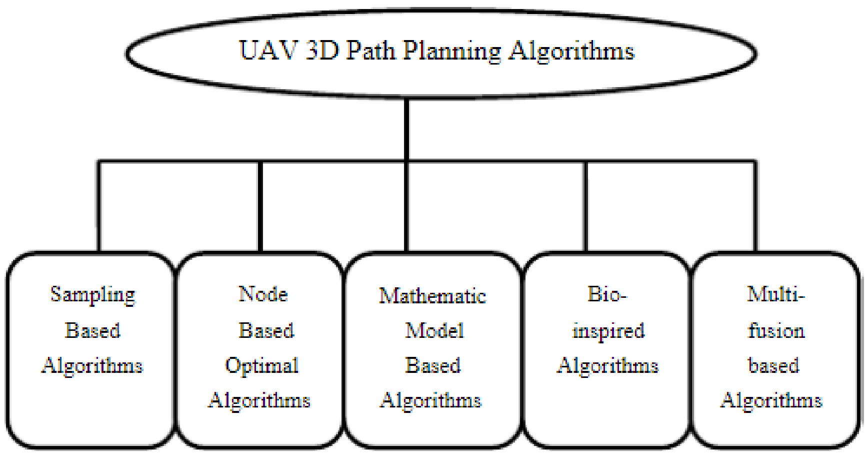

UAV 3D Path Planning Algorithms
Sampling Based Algorithms
Node Based Optimal Algorithms
Mathematic Model Based Algorithms
Bio-inspired Algorithms
Multi-fusion based Algorithms

Figure 1. Path Planning Approaches [22]

The sampling-based algorithms break into two sub-categories; active and passive algorithms. Active algorithms include Rapidly-exploring Random Trees (RRT), which can build a framework to the goal within its own processing procedure. Whereas passive algorithms build a number of acceptable paths, but do not choose a final answer, requiring another algorithm to determine optimality [22]. The options in this category include passive elements like 3D voronoi , Rapidly-exploring Random Graph, probabilistic roadmap (PRM), Kinetic-PRM, Static-PRM, Visibility Graphs, and Corridor Map. Active options includes elements such as RRT, Dynamic Domain RRT(DDRRT), RRT-Star(RRT*), and Artificial Potential Field [22]. All sampling-based algorithms require some prior information regarding the workspace.

The “Node Based Optimal Algorithms” approach is similar to sampling-based methods, in that they are dependent on the system to sense the area ahead of time and perform some post processing on that data to create nodes and arcs. Examples of this approach include; A*, Lifelong Planning A*, Dynamic A*, D*-Lite, and others [22]. After the construction of the nodes and arcs, paths are compared against a cost function, to determine the optimal path [22]. These algorithms have been demonstrated on UAVs previously, including [23]. D* is by far the most popular of these methods and would be a strong contender for this CONOPs.

The “Mathematical Model Based Algorithms” are best recognized as Linear Programming (LP) and Optimal Control algorithms. The benefits of these methods are that they not only path plan, but take into account the environment and the body, evaluating both the kinematic and dynamic constraints of the system and use these to bound the cost function with inequalities to find the optimal state. The issue with these approaches is the high computational cost associated with evaluating both the environment and vehicle limits as variables [22].

The “Bio-inspired Algorithms” are a family of algorithms based on mimicking biological behavior. Examples of these include evolutionary algorithms and neural networks. Evolutionary algorithms further break down into genetic algorithms, memetic algorithms, particle swarms, and colony optimization [22]. The basic principle of each of these approaches is that they start by introducing a variety of paths, and begin evaluating the best of each, taking the best from each run then making slight course changes and re-evaluating (mimicking genetic mutation and evolution). The result is a path likely to be very near optimal, as mutations are introduced to overcome local minima, but premature convergence can remain an issue.

The “Multi-Fusion Based Algorithms” are exactly as they imply. By taking several separate algorithms and marrying them in order to find true global optimality. Higher fidelity solutions have been created such as in [24], which used 3D grid for the environment and 3D PRM to form an obstacle free road map, then finally applied A* to find the optimal path. Other examples exist as well, such as [25], which combined visibility graphs and a Dijkstra's algorithm. These fusions approaches can significantly improve the optimality of the path found but require varying amounts of additional computational power to solve, so care must be taken to ensure the algorithms used are within the limits of a UAV companion computer.

### 2.2.6 Real Time Kinematic (RTK) GPS Position Holding

The Global Navigation Satellite Systems (GNSS) provide a variety of positioning state solutions. These are single point positioning (SPP), precise point positioning (PPP), differential GPS (DGPS) and real time kinematic (RTK). The differences between these solutions have to do with types of measurements they take, the data epochs required, and the number of receivers involved in the positioning operations. The utility of each of these solutions is highly dependent on the application demands and environment.

SPP uses a single receiver and epoch to create a pseudo-range measurement. PPP solutions require both phase measurements and code from a single receiver, but require a long period of observations. DGPS solutions are based on code measurements from a single epoch but use differential corrections from a reference station or network. Finally, RTK positioning uses the carrier phase measurements in the DGPS mode, preferably from a single epoch (or at least a short period of time [26]). RTK, when fully surveyed in, can provide locational accuracy on the level of centimeters [26], and with a proper base station broadcast power can inform nearby

RTK receivers out to distances beyond 50-70km to accuracies within 10cm north and east and 30cm up and down [26].

The use of RTK on UAVs has been demonstrated in a variety of fields including coastal surveying [27] and precision agriculture [28]. In these applications, the ability of the vehicle to know with high accuracy its current location, enables all other sensing and surveying tasks. The existence of RTK solutions for mining and precision agriculture on the ground has existed for years, and would cost on the order of 30 to 50 thousand dollars for the base stations and receivers. Only recently has the price point dropped to a level and size that hobbyists can make use of RTK with systems like the Here+ system ($600). These new, affordable, systems are bringing RTK to the masses, but lack some of the range and consistency of the large industrial systems.

## 2.3 Sensors Review

Environmental sensing for robots has been a major area of research for many decades. In the area of UAVs there are a few stand-outs for their weight, power, and accuracy. These include vision-based systems, which are utilized for visual odometry (VO), or optical flow, sonic based systems, which return distance measurements using sound returns off surfaces, and laser-based systems which use light returns to determine distance, and sometime angles.

Vision based systems like the one demonstrated in [29], and covered in great detail there, make use of monocular or stereo vision to detect features in frames and track them to the next. A VO or optical flow algorithm then deciphers these changes and can then be used to produce a 3D model of the area or detect objects on trajectory to block the path of the vehicle. The issue with VO is that it is a computationally intensive process, requiring the evaluation of hundreds or thousands of separate features, their relative movement, and disappearance. Optical flow is a lighter algorithm, but as discussed previously, depends on the acuity of the sensor and various algorithms to track motion.

Ultrasonic based systems utilize echo-location principles to detect and report the existence and distance to objects in their field of regard. The basic principles of this are demonstrated in [30]. As described therein, the sensors report back the nearest distance detected, and as such require a constellation of sensors to fully comprehend the surroundings of the UAV. The result is a large number of individual sensors pointed in an array around the vehicle, and an algorithm designed to understand and react to those measurements. The drawback to this approach is dependent on the number of sensors required; in [30] that number is 12. This provides many potential sources of failure, extra power draw, and weight.

Finally, laser-based systems, such as LIDAR, utilize light reflection signatures to determine distance. Again, many examples of these systems are used in research. In [12], a 2D LIDAR is rotated to create a 3D map of the environment around the vehicle. In [13], a LIDAR system is utilized in an obstacle avoidance and detection system, much like the system required in the CONOPS. LIDAR systems work by spinning an optical element in front of a laser and a light sensor. The optical element is on a servo that sweeps between the angles defined for the system, and at each increment of servo turn, the sensor detects the distance between it and a surface, this distance is then married to the angle at which it was registered. The result is a point cloud of angles and distances around the LIDAR.

The set-backs for these systems are the higher power requirements for meaningful distance detection on UAVs and weight.

## 2.4 Summary

In this chapter we reviewed the wide range of research related to the CONOPS. We discussed the various means of accomplishing obstacle avoidance, including optical flow, and LIDAR based systems. We discussed wall-following algorithms and looked at examples of each. Finally, we reviewed the many path planning algorithms that exist, and compared the strengths and weaknesses of each. Based on these findings, it was determined that the wall-following portion of the CONOPS could best be demonstrated using LIDAR. This is as a result of the breadth of the regions a single sensor could keep track of, and the ability to measure angles, thus allowing yaw adjustments to be included in the algorithm. This is important as the direction the mission sensor points is crucial for keeping it in its operating region and gathering operator directed data from specific places. Additionally, RTK GPS will be tested to provide an initial understanding of the capability of that GPS solution to keep a multi-rotor within a certain location. RTK GPS presents a potential for removing much of the sensor overhead requirements in GPS permissive environments.

# III. Architecture

## 3.1 Introduction

This research provides an example of how low-cost sensor systems, such as the LIDAR utilized in this thesis, can provide immediate, useful, and efficient improvements to building proximity flight systems. The architecture presented will define the full range of required activities necessary for operating a UAV near a building with a unidirectional mission sensor. This architecture will include aspects not investigated in this thesis, but which would be required to meet the full CONOPs of the research sponsor. Finally, the portions of the architecture which were researched included a LIDAR sensor which allowed for accurate predictable flight near building faces, utilizing a simple logic algorithm, and positional holding accuracy using an RTK-GPS. This approach has the potential to be further expanded to handle more complex building structures and surface types, especially with the inclusion of more than one type of sensor.

## 3.2 Overview

In the previous chapter, the use of a variety of algorithms were reviewed to determine the most immediately relevant type for meeting the needs of the CONOPs. In conjunction with this, a variety of distance and location sensors were reviewed for their ability to inform the algorithm, while minimizing the size, weight, and power requirements necessary to small tactical UAVs. This chapter will detail the architecture of the system, and the means by which this combination of hardware and code were prepared, assembled, and evaluated.

## 3.3 Architecture

### 3.3.1 Introduction

In the following section an explicit instance of the following architecture will be described, but it is not the only possible instance. It will be demonstrated that the necessary elements of the system were met with this specific construction, but many other possible variations might be designed depending on differences from environmental requirements and structure types, or mission sensor sizes and power needs.

### 3.3.2 Development

The architecture was developed by constructing a Concept of Operations (CONOPS) with the research customer and verifying that all necessary requirements were captured for the system. This CONOPS (Appendix A) helped to properly scope the effort to the true needs of the customer and avoid introducing unnecessary complexity (and cost) into the design. The CONOPS emphasized the importance of limiting the control input by the operators while they were operating the mission sensor. This necessitated a control algorithm and architecture which could feed environment information to the algorithm by some means of on-board sensing.

In addition to reactionary algorithms, a means of determining location relative to the building accurately will be required. This research falls outside scope of this thesis for demonstration, but it has been included in the architecture as a higher level of spatial awareness is necessary for intelligently navigating and targeting specific areas of a structure of interest. The requirement for this level of awareness becomes clear when considering the importance of being able to know what has been viewed by the mission sensor and what still requires examination.

The operational activity diagram of the system is shown in Figure 2, and the full operational activity model is shown in Figure 3. This diagram captures the full breadth of activities that will be required of the system for the successful completion of its CONOPs. Basic operations already encompassed in the PixHawk2 autopilot functions like waypoint following, vehicle status info, and signal strength reporting require no further study or research and have been demonstrated in other projects. The unique portions of this architecture are captured immediately following the changing of flight modes beside the target structure. These are the "wall following", "localization and positioning", "mission sensing and data streaming", and "path finding" activities. In Figure 3, these are noted with a red square. These functions make up the unique capabilities which would support a CONOPs in line with the structure sensing mission.

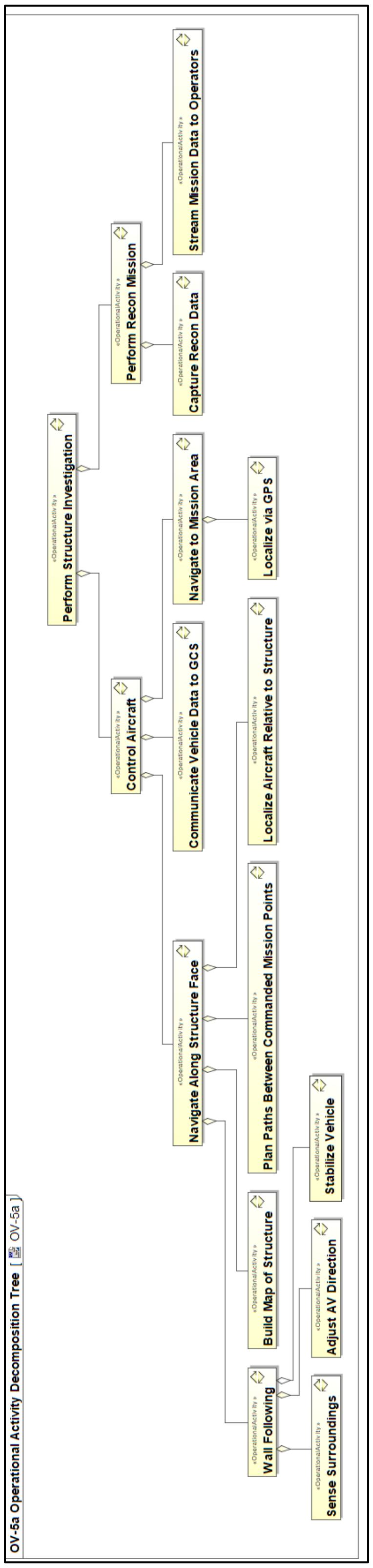
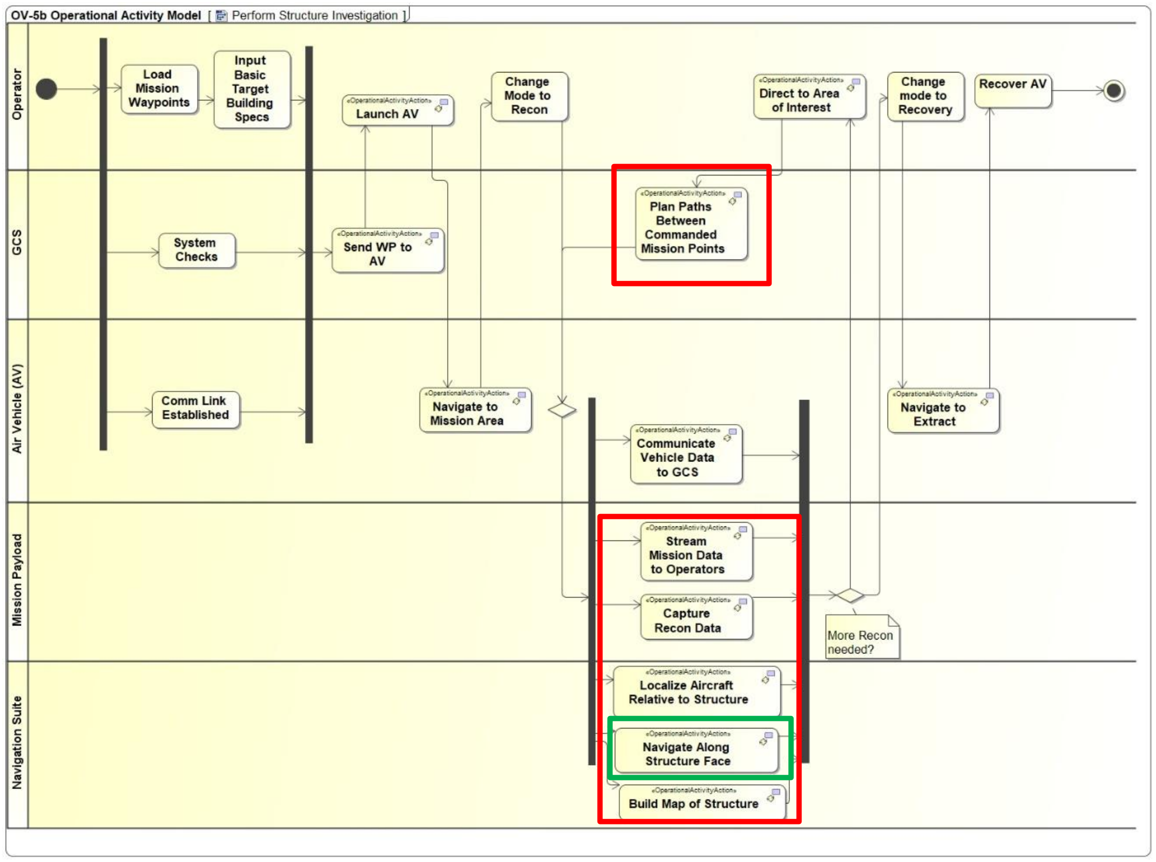

OV-5a Operational Activity Decomposition Tree [ OV-5a ]
«OperationalActivity»
Perform Structure Investigation
Control Aircraft
Perform Recon Mission
Capture Recon Data
Stream Mission Data to Operators
Navigate to Mission Area
Localize via GPS
Communicate Vehicle Data to GCS
Localize Aircraft Relative to Structure
Navigate Along Structure Face
Plan Paths Between Commanded Mission Points
Build Map of Structure
Wall Following
Sense Surroundings
Adjust AV Direction
Stabilize Vehicle

**Figure 2. Operational Activity Tree**



OV-5b Operational Activity Model [ Perform Structure Investigation ]
Operator
GCS
Air Vehicle (AV)
Mission Payload
Navigation Suite
Load Mission Waypoints
Input Basic Target Building Specs
Launch AV
Change Mode to Recon
Direct to Area of Interest
Change mode to Recovery
Recover AV
System Checks
Send WP to AV
Plan Paths Between Commanded Mission Points
Comm Link Established
Navigate to Mission Area
Communicate Vehicle Data to GCS
Navigate to Extract
Stream Mission Data to Operators
Capture Recon Data
More Recon needed?
Localize Aircraft Relative to Structure
Navigate Along Structure Face
Build Map of Structure

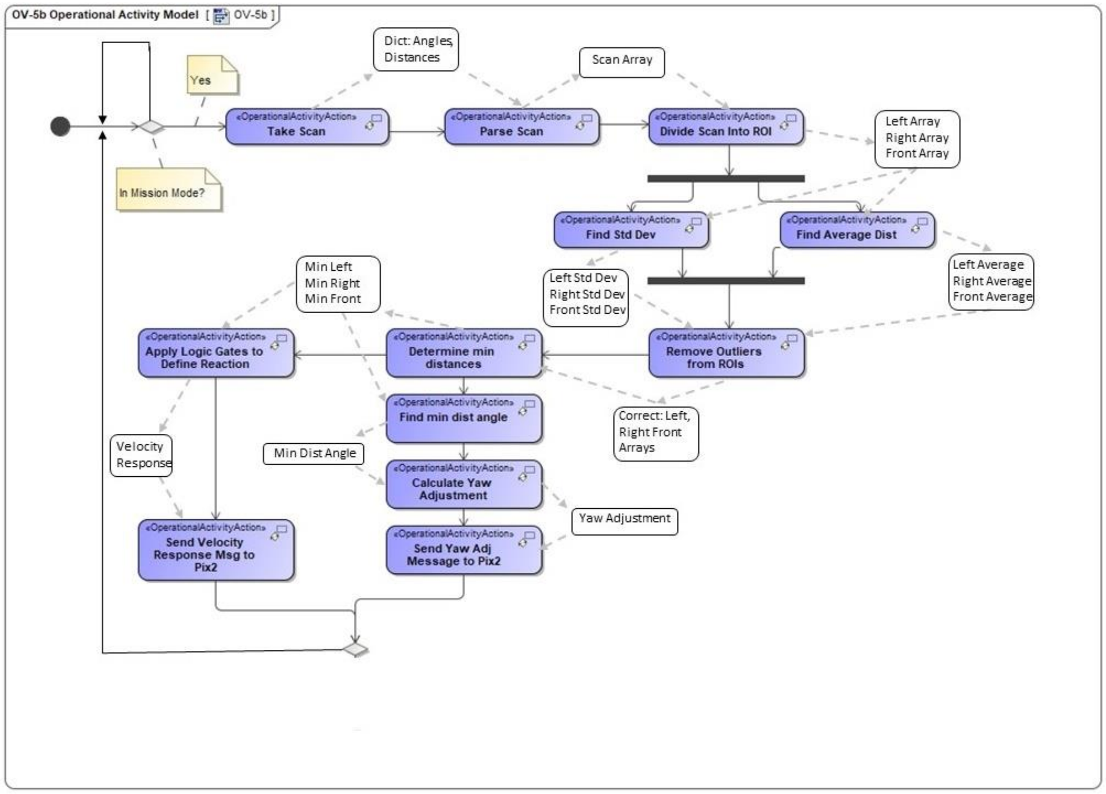
**Figure 3. Full System Operational Activity Diagram**

The mission sensing piece is a straight forward piece of this architecture. This is the reason why the vehicle needs to be flown in proximity to the structure. The operating parameters of this system will dictate the operating conditions for the rest of the system. Parameters like top speed, max/min distances from target, allowable rate of distance variance, and power demands will all have significant impacts on the vehicle required and operations of the other portions of the architecture, as this is the primary functional focus of the system. There would be little gained by optimizing the system to wall following or path finding activities if they came at the expense of the mission sensing capability. This portion of the architecture will not be demonstrated in the thesis, as the equipment necessary is still in development and not required for performing the flight near structures portion of this thesis.

A Simultaneous Localization and Mapping (SLAM) algorithm or Visual Odometry (VO) algorithm capability will be required to provide accurate awareness of where the vehicle is in relation to the larger structure. A system which is simply initiated and forgotten provides a minimal operational benefit when attempting to investigate specific places and floors on the structure; therefore, a requirement that the system be capable of relating its location relative to the target structure is necessary.  There are a number of possible ways to accomplish this task, two of which are SLAM and VO, but these are not the only approaches. Those two are suggested solely based on the maturity and detailed level of research which exist, as referenced in Chapter II, for those methods of localizing a vehicle to its surroundings using local data cues (not at all or only partially referencing GPS). Both of these approaches are computationally intensive, especially VO, and are likely to require off-board processing. This portion of the architecture will not be demonstrated in the data collection for this thesis.

A path finding algorithm will be necessary around a building, especially as the complexity and shape of the structure changes. If the vehicle is ordered to move from its current location to a higher priority location, an algorithm would be necessary to determine the fastest/lowest risk path to that location, as a direct path can’t be assumed available. It would be expected that this path finding algorithm would piggyback on the localization and mapping work being performed in parallel with the other capability. A variety of path finding algorithms exist, and they can be chosen to prioritize fastest routes vs time to process. Examples of optimization-based path finding that might be used are detailed in [31]. This portion of the architecture will not be demonstrated under this thesis but would be highly recommended for follow-on efforts.

Finally, the wall following portion of the architecture is simultaneously the most mature and central to the overall architecture. This capability allows the vehicle to remain within the operating parameters of the mission sensor while navigating safely around the target structure with minimally aggressive control inputs to keep it there. More will be discussed regarding the specific code created to accomplish this later in the chapter (3.4.2.1), but as the demonstrated portion of this thesis it will receive more attention than the other three major activities in the architecture. The activity diagram related to the wall following sub-portion of the architecture is shown in Figure 4.



OV-5b Operational Activity Model [ OV-5b ]
Yes
In Mission Mode?
Dict: Angles, Distances
Scan Array
Take Scan
Parse Scan
Divide Scan Into ROI
Left Array Right Array Front Array
Find Std Dev
Find Average Dist
Left Std Dev Right Std Dev Front Std Dev
Left Average Right Average Front Average
Min Left Min Right Min Front
Apply Logic Gates to Define Reaction
Determine min distances
Remove Outliers from ROIs
Find min dist angle
Correct: Left, Right Front Arrays
Min Dist Angle
Velocity Response
Calculate Yaw Adjustment
Yaw Adjustment
Send Velocity Response Msg to Pix2
Send Yaw Adj Message to Pix2

**Figure 4. Wall Following Activity Diagram**

The full architecture required for a mission sensor requiring near structure maneuvers and situational awareness is dependent on three main tasks. The first is safe operation near the structure face; this is accomplished most effectively using a wall following algorithm informed by distance and direction sensors. This capability will be further investigated and demonstrated in this thesis. The second is localization and mapping of the operating region relative to the air-vehicle. This is crucial to performing efficient surveying of a structure and being able to direct the mission sensor to places of immediate interest, as that is impossible without first understanding where the vehicle is in relation to that point. Finally, a path finding algorithm is necessary to make use of the mapping performed by the previous system to allow the vehicle to accurately and efficiently proceed from its current location to an operator defined mission interest.

## 3.4 Wall Following

### 3.4.1 Introduction

The operation of wall following can be accomplished in a variety of ways; for this thesis the operation will be performed primarily by ingesting LIDAR sensor data, and categorizing the ranges into specific reactions based on pre-determined angle and range gates. This approach is considered a logic gate/decision tree approach and is in theory a simple means of evaluating data and creating a consistent and predictable output. What follows are the specific choices and gates chosen for the "Wall Following" operation.

### 3.4.2 Code

#### 3.4.2.1 Code Architecture

A decision tree model, based on [32] and other wall following algorithms, was selected for the approach of this project. This decision tree is built using logic gates which evaluate the relevant parameters related to the control and state of the vehicle and output a new velocity vector to respond to that state. In the case of this thesis, the LIDAR is sampling the environment state of the vehicle and returning distances and angles from the body to the companion computer. The companion computer takes these values and manipulates them through a series of functions to define the regions of interest (ROI) shown in Figure 5, which inform the control logic. The ROI are themselves divided into zones; Zone 1-4 and Danger Zone. These zones are defined by the operator, and determine the operational clearances of the vehicle, and provide the framework for the decision tree.

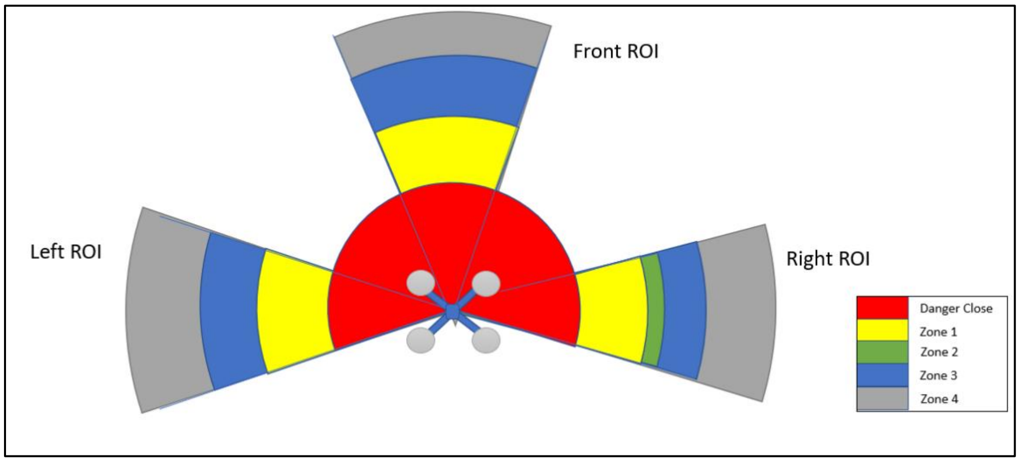
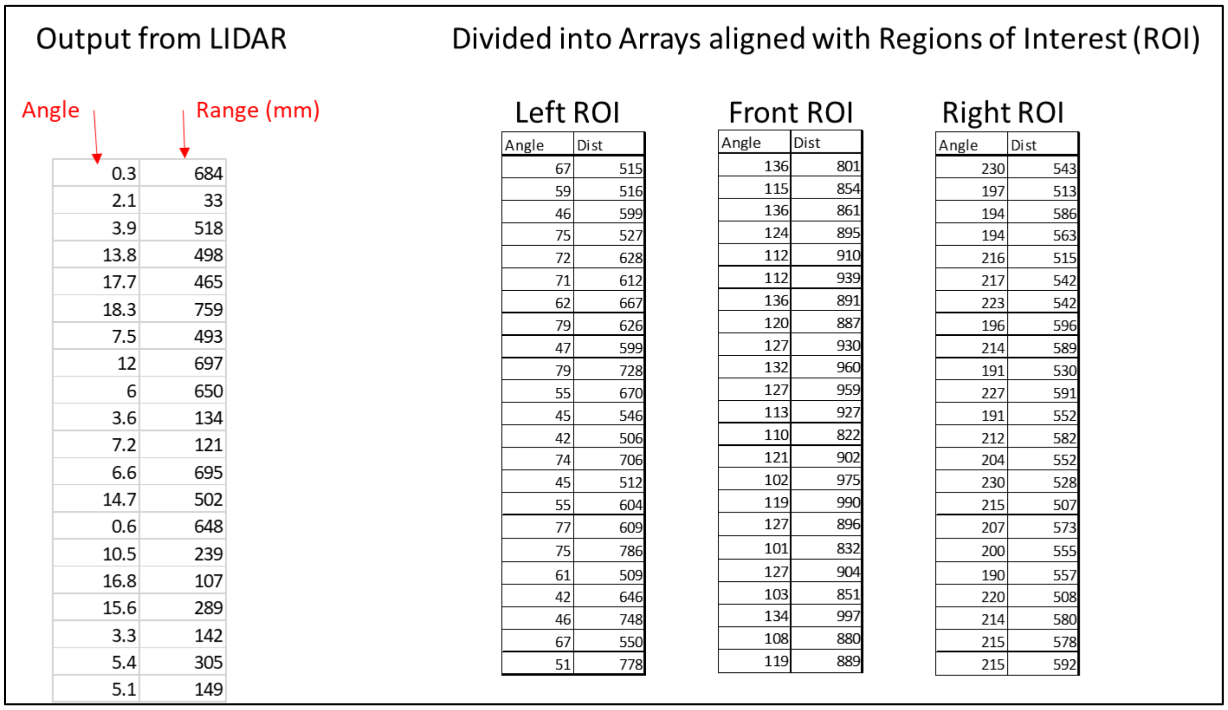


**Figure 5. LIDAR Regions of Interest**

The output from the LIDAR itself resembles Figure 6. That “dictionary” array is then sub-divided into ROIs, as shown in Figure 6, using the “numpy.where”, function which allows a table to be sorted and divided using logic arguments. Once these ROIs are created in array form, they are further gated to remove single step outliers and deal with the “0 distance” for no return issue. This is accomplished by first taking the average of a given ROI, finding the standard deviation, then creating a new ROI array omitting any results that fall outside that standard deviation of the mean, as demonstrated in Figure 7. With this new array, the minimum distance is recovered, if this minimum distance is again zero, or within the new standard deviation of the array from zero, the code returns the outer threshold distance plus 100mm. Otherwise, that min distance is the value used in the decision tree for that ROI.


Output from LIDAR
Divided into Arrays aligned with Regions of Interest (ROI)
Angle
Range (mm)

| Angle | Range (mm) |
|---|---|
| 0.3 | 684 |
| 2.1 | 33 |
| 3.9 | 518 |
| 13.8 | 498 |
| 17.7 | 465 |
| 18.3 | 759 |
| 7.5 | 493 |
| 12 | 697 |
| 6 | 650 |
| 3.6 | 134 |
| 7.2 | 121 |
| 6.6 | 695 |
| 14.7 | 502 |
| 0.6 | 648 |
| 10.5 | 239 |
| 16.8 | 107 |
| 15.6 | 289 |
| 3.3 | 142 |
| 5.4 | 305 |
| 5.1 | 149 |

Left ROI

| Angle | Dist |
|---|---|
| 67 | 515 |
| 59 | 516 |
| 46 | 599 |
| 75 | 527 |
| 72 | 628 |
| 71 | 612 |
| 62 | 667 |
| 79 | 626 |
| 47 | 599 |
| 79 | 728 |
| 55 | 670 |
| 45 | 546 |
| 42 | 506 |
| 74 | 706 |
| 45 | 512 |
| 55 | 604 |
| 77 | 609 |
| 75 | 786 |
| 61 | 509 |
| 42 | 646 |
| 46 | 748 |
| 67 | 550 |
| 51 | 778 |

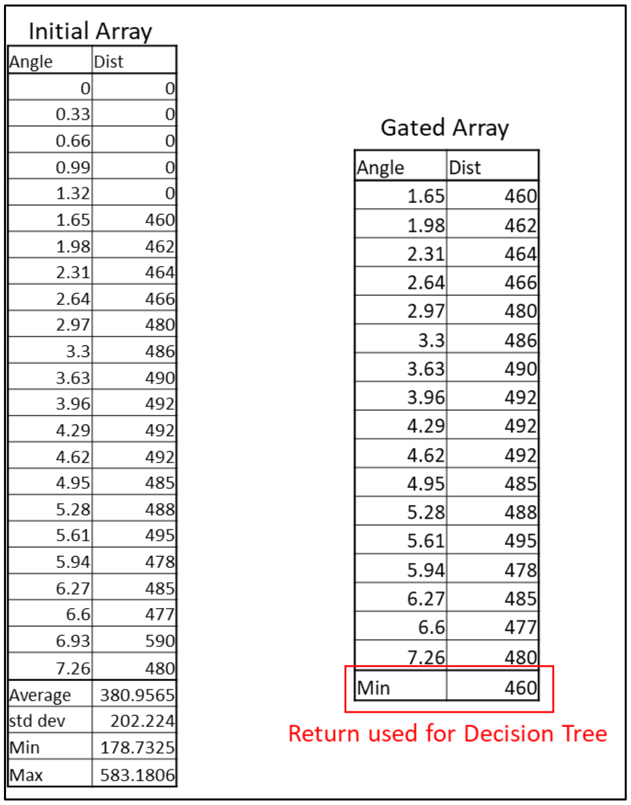
Front ROI

| Angle | Dist |
|---|---|
| 136 | 801 |
| 115 | 854 |
| 136 | 861 |
| 124 | 895 |
| 112 | 910 |
| 112 | 939 |
| 136 | 891 |
| 120 | 887 |
| 127 | 930 |
| 132 | 960 |
| 127 | 959 |
| 113 | 927 |
| 110 | 822 |
| 121 | 902 |
| 102 | 975 |
| 119 | 990 |
| 127 | 896 |
| 101 | 832 |
| 127 | 904 |
| 103 | 851 |
| 134 | 997 |
| 108 | 880 |
| 119 | 889 |

Right ROI

| Angle | Dist |
|---|---|
| 230 | 543 |
| 197 | 513 |
| 194 | 586 |
| 194 | 563 |
| 216 | 515 |
| 217 | 542 |
| 223 | 542 |
| 196 | 596 |
| 214 | 589 |
| 191 | 530 |
| 227 | 591 |
| 191 | 552 |
| 212 | 582 |
| 204 | 552 |
| 230 | 528 |
| 215 | 507 |
| 207 | 573 |
| 200 | 555 |
| 190 | 557 |
| 220 | 508 |
| 214 | 580 |
| 215 | 578 |
| 215 | 592 |

**Figure 6. LIDAR Output and ROI Segregation**



Initial Array

| Angle | Dist |
|---|---|
| 0 | 0 |
| 0.33 | 0 |
| 0.66 | 0 |
| 0.99 | 0 |
| 1.32 | 0 |
| 1.65 | 460 |
| 1.98 | 462 |
| 2.31 | 464 |
| 2.64 | 466 |
| 2.97 | 480 |
| 3.3 | 486 |
| 3.63 | 490 |
| 3.96 | 492 |
| 4.29 | 492 |
| 4.62 | 492 |
| 4.95 | 485 |
| 5.28 | 488 |
| 5.61 | 495 |
| 5.94 | 478 |
| 6.27 | 485 |
| 6.6 | 477 |
| 6.93 | 590 |
| 7.26 | 480 |
| Average | 380.9565 |
| std dev | 202.224 |
| Min | 178.7325 |
| Max | 583.1806 |

Gated Array

| Angle | Dist |
|---|---|
| 1.65 | 460 |
| 1.98 | 462 |
| 2.31 | 464 |
| 2.64 | 466 |
| 2.97 | 480 |
| 3.3 | 486 |
| 3.63 | 490 |
| 3.96 | 492 |
| 4.29 | 492 |
| 4.62 | 492 |
| 4.95 | 485 |
| 5.28 | 488 |
| 5.61 | 495 |
| 5.94 | 478 |
| 6.27 | 485 |
| 6.6 | 477 |
| 7.26 | 480 |
| Min | 460 |

Return used for Decision Tree

**Figure 7. Data Gating Example**

Those regions are then evaluated using a decision tree, as shown in Figure 8 to provide the corresponding output velocity for the vehicle. These outputs were defined by carefully evaluating all potential state measurements of the vehicle and determining acceptable responses to those situations. For this thesis a matrix, shown in Table 1, of some possible variants of measurements was built, the full matrix is available in Appendix C, with a descriptive statement regarding the likely scenario in which that might be encountered, followed then by the proper velocity response to that situation.

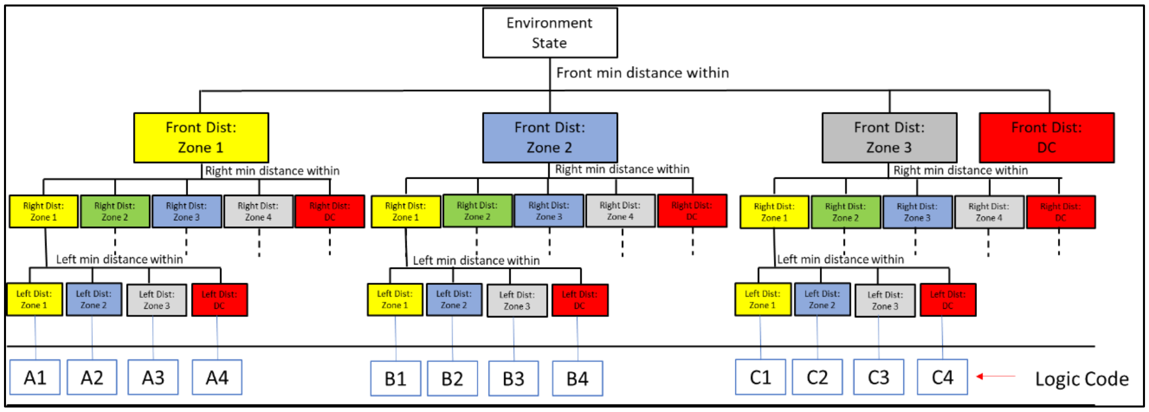


**Figure 8. Decision Tree Used for Logic Code**

**Table 1. Logic Table Portion**

| Front Measurement | Right Measurement | Left Measurement | Scenario | Reaction | Code | Forward | Right | Yaw |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | End of tight corridor | Stop. Turn 180. Forward 2 sec @ half speed | A1 | 0 | 0 | 180 |
| 1 | 3 | 2 | End of close corridor | Stop. Turn 180 | A2 | 0 | 0 | 180 |
| 1 | 3 | 3 | At corner | Stop. Turn left 90 | A3 | 0 | 0 | Ccw @ rate |
| 1 | 3 | DC | Dangerously close on left, close corridor | Stop. Crab right and backwards | A0 | -0.5 | 0.5 | 0 |
| 1 | 4 | 1 | Tight corridor, end, open right | Stop. Turn right 90 | A4 | 0 | 0 | cw @rate |
| 1 | 4 | 2 | Close corridor end, open right | Stop. Turn right 90 | A5 | 0 | 0 | cw @rate |

One danger of using this type of logic gate decision tree is that the vehicle can enter states on the verge of two different responses and begin making abrupt course and velocity changes; this is referred to as “logic lock”. In order to avoid such scenarios, many of the responses are built from curves based on the proximity to those gates, making the responses nearest a gate more limited than those falling far outside those boundaries. For example, the velocity correction for moving the vehicle closer to the building face if it begins to drift out of the desired distance, is shown below in Equation 1.

$$Velocity = (Currentdist(mm) - Optimumdist(mm)) * (AdjFactor) \quad (1)$$

Through a combination of these types of response functions, and creating gates dependent on vehicle parameters, the overall system gains a level of adaptability and finesse that will support broader applications.

#### 3.4.2.2 Code Testing

The code developed for this thesis is based on a variety of prior systems. The overall intent of testing will be to tune the logic gates and response curves to best match our system. Based on the testing some broad level tuning may be created, but it is likely that individual systems will need to have the response curves adjusted based on the response parameters of their autopilot and system dynamics. That said, here is the test regime executed by the thesis vehicle.

The first step was basic table top system tests. The table top tests were used to confirm that the code was outputting the correct responses to designed scenarios. A matrix of tests using this set-up is shown in Figure 9. Once these parts of the code were tuned acceptably, and the LIDAR/Linux interaction was stable, the testing entered the next phase.

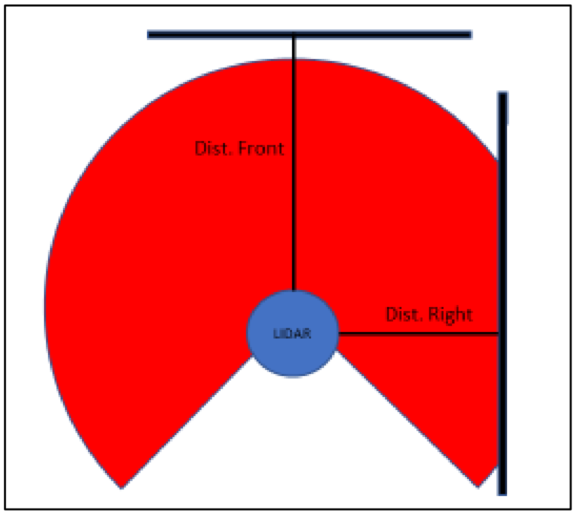


**Figure 9. Table Top Test Example**

The second step of the testing was cart testing. Cart testing was used to confirm that the output from the companion computer reacted quickly and accurately to changes in the environment by entering different states sequentially. These tests further stabilized the code design and are likely to be the final extent to which the code can be closely ported between dissimilar systems. Each region has a specific reaction anticipated from the algorithm, and confirming those reactions, at acceptable rates, is necessary for progressing to vehicle testing.

The next step of testing was cage testing. These tests provided the first glimpse into the full system directly controlling a vehicle autopilot and real system responses. These tests confirmed the success of the vehicles integration between the code, hardware, and sensor systems. This required a "build up" approach that started with very basic maneuvers to confirm full system stability, and cart testing to confirm system response capabilities.

#### 3.4.2.3 RTK GPS Testing

The evaluation of RTK GPS as a means of holding close positional tolerances with a small multi-rotor was necessary to determine the necessity of the environment sensor payload, especially in GPS permissive environments. Characterizing the level of precision possible with an RTK solution in a small UAV sized system will allow future users to understand the upper bounds of such a system.

This evaluation was accomplished by bringing the multi-rotor up to a specific height, giving it a waypoint (location and altitude) with zero radius, and telling it to hold on that waypoint. The autopilot would then use its RTK fix to attempt to maintain the vehicle in that position against any drift and breeze interference. The photo measured vehicle location was then compared against the vehicles RTK measured location to provide an overall vehicle location holding accuracy, yaw variance, and altitude variance. The setup for said testing is depicted below in Figure 10.

This test setup was operated by an intervalometer and programmed to take 45 pictures at a rate of 1 per second. The result being 45 directly measurable location, yaw, and altitude variances from each test. These results will be discussed in 4.2.

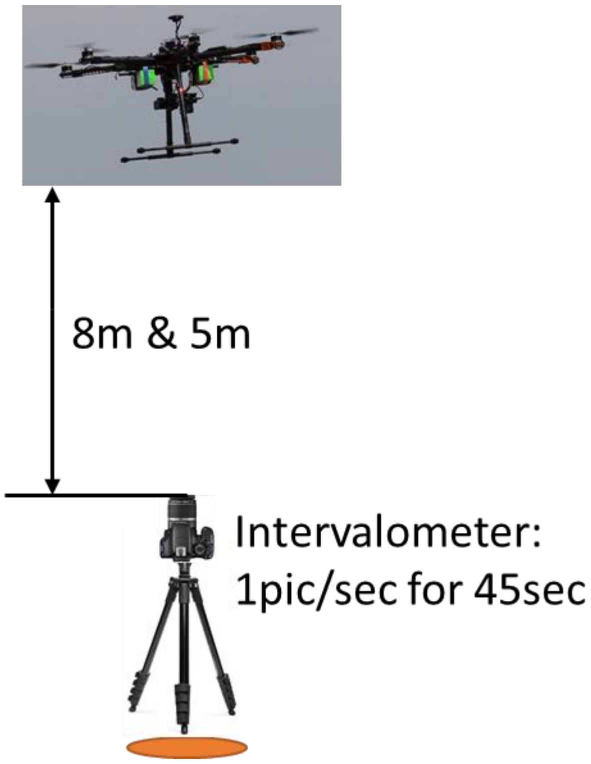


**Figure 10. RTK GPS Test Set-up**

### 3.4.3 Hardware

The hardware used in this thesis were based on a variety of previous projects in the AFIT ENV team. The system can be broken down into three separate component systems; the air vehicle, the autopilot system, and the wall following system.

The air vehicle is based on a Tarot T960 hex-frame multi-rotor. Composed largely of carbon fiber components and structural members it is very light for its size and rigid enough for significant payloads. Figure. below shows a picture of the vehicle used. The motors are KDE Direct 425Kv, and the speed controllers are 40A. This is the exact same specification as was used in the AFIT CE runway rapid assessment project in 2017. The hex frame design provides significant stability and power, while taking minimal impacts for control-ability and endurance.

The autopilot system utilized on this airframe is a COTS PixHawk2 (Pix2) with a Here+ GPS system. These autopilots have built in algorithms for handling everything from fixed wing vehicles through ground and multi-rotor configurations. In addition to this, the Pix2 is designed to handle direct inputs from external scripts that are sent using MAVLink protocols. This allows the rest of the system to direct the control of the air-vehicle within the Pix2's well established control PIDs and algorithms. A Pix2 is shown in Figure 11.

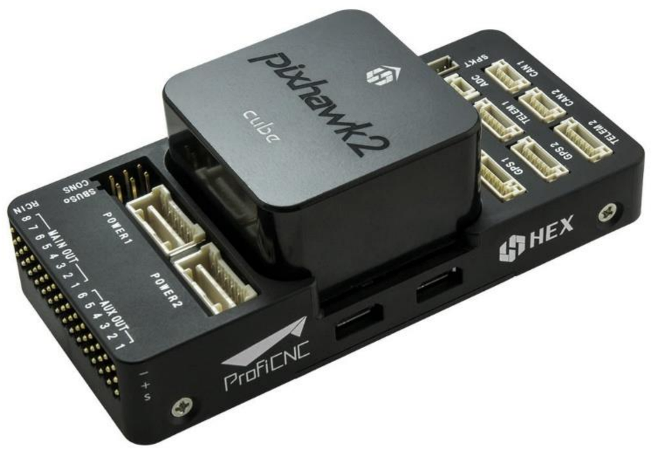


**Figure 11. Pixhawk 2 Autopilot**

Finally, the sensing system and companion computer running the Python script are a Hokoyu URG-04LX LIDAR system, and a BeagleBone Black (BBB) processing board running Debian Ubuntu. These components in the system perform all the upstream measurement, parsing, logical evaluation of the LIDAR data, and sending of the MAVLink messages to the Pix2. A figure of the BBB and LIDAR system are shown below in Figure 12.

**Figure 12. BBB and LIDAR**

The Wall Following capability demonstrated in this thesis is only one build out of an architecture that might have better sensing options and more comprehensive algorithms depending on on-board processing power available, structure face materials, and vehicle design. The choices made for this thesis represent the best combination of attributes for the demonstration/validation of the concept.

## 3.5 Chapter Summary

This chapter reviewed the full complexity and capabilities necessary to the structure proximity flight operation. Three central capabilities were defined as the minimum necessary to accomplish useful and safe sensing of a structure with a proximity air vehicle. Finally, the specific portion of that architecture being investigated by this thesis was defined and the methodology of its creation and testing were laid out. The next two chapters will review the results of those investigations.

# IV. RTK GPS Results and Discussion

## 4.1 Chapter Overview

This chapter will present the results of the RTK-GPS position hold tests. An evaluation of RTK as a potential means of path following near a structure, and detail potential drawbacks and yet to be investigated concerns regarding the use of RTK in this role.

## 4.2 Results

The testing of the RTK GPS, in the manner defined in 3.4.2.3, provided a very useful set of data for understanding the overall performance of a small UAV married to one of the new affordable RTK GPS solutions, the Here+ antenna and RTK base station.

The measurements used for the location data were captured in the following way. Each of the 45 photos taken had the midpoint of the vehicle identified as shown in Figure 13; this pixel location was recorded, with the same process applied to the next sequential photo. Given the known size of the vehicle, a cm/pixel conversion could be derived and used to find the location variation between captures. These tests were performed at an altitude of 5m and 8m.

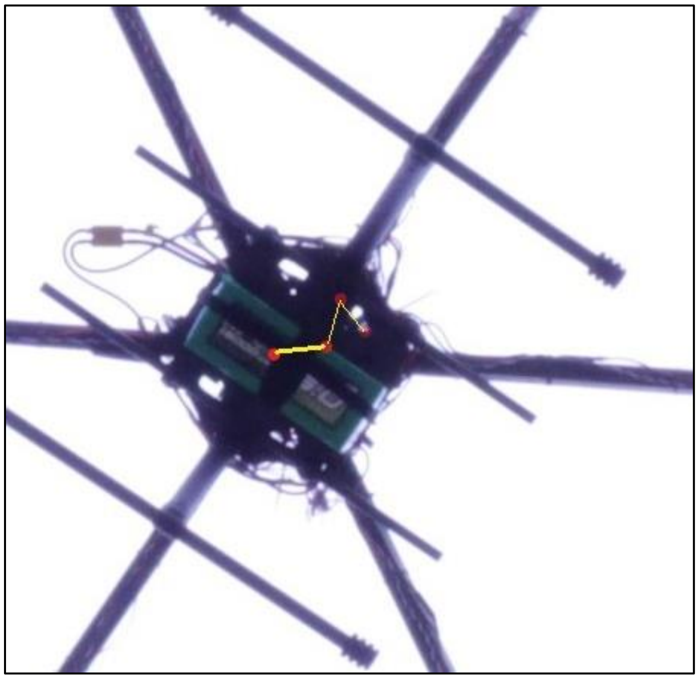
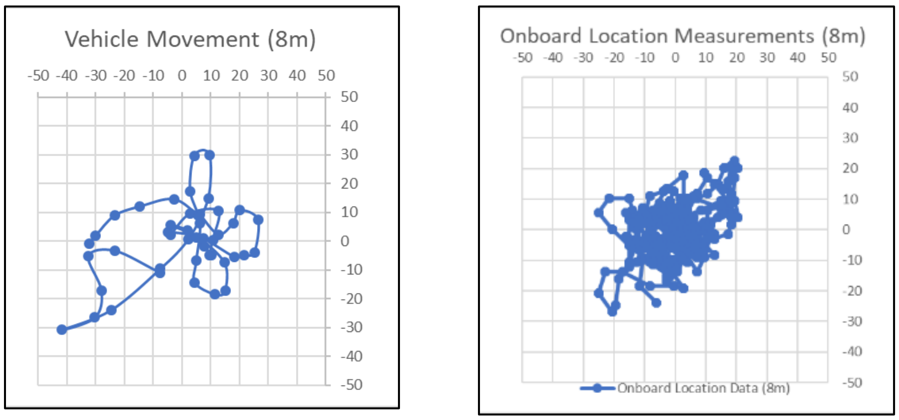


**Figure 13. Location Measurement Example**

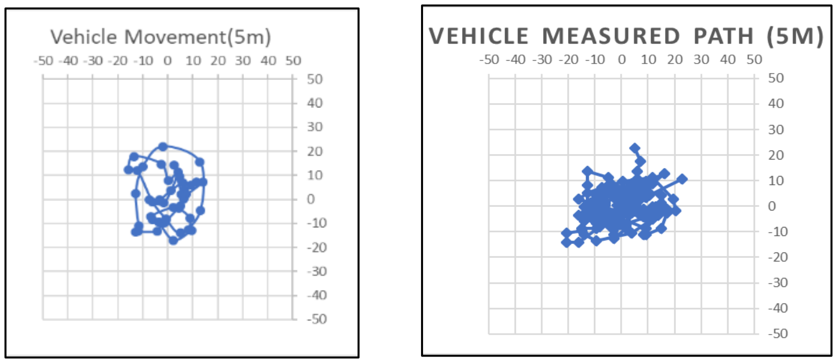
The location variation, as shown in Figure 15 and Figure 14, and its measured variance, gives a good overall impression of the capability of a UAV to maintain position with the Here+ RTK GPS system. It should be noted that the sample rate of the real test was only 1Hz, while the on-board system samples at about 3Hz. Both of these tests showed an overall location holding accuracy that varied slightly. In one test, the vehicle remained within a 40 x 40cm box. In the second test, that box expanded to a 70 x 60 cm box. The measured standard deviations are 8.8 x 10.1cm and 17.0 x 12.7cm respectively. When compared to the stand-alone GPS positional accuracy of ~2.8m on FAA receivers [33], these accuracies demonstrate a clear improvement for position holding overall for any vehicle using GPS as the main source of its positional data. The cause of the inconsistencies between the two tests is inconclusive, but both still show a marked improvement over the baseline SPP GPS/IMU fusion. It may be that the higher altitude of the 8m test was subject to stronger winds than the 5m test, but that was not confirmed by measurement, and weather reports for the day recorded only calm weather during testing periods.



**8 meter altitude test**

- Internal GPS measured
    - Std dev: 9.3cm x 8.3cm
- Photo Analysis
    - Std dev: 17cm x 12.7cm

**Figure 14. Test 2 Real (L) vs Onboard (R) Location Measurement**



**5 meter altitude test**

- Internal GPS measured
    - Std dev: 8.8cm x 6.2cm
- Photo Analysis
    - Std dev: 8.0cm x 10.1cm

**Figure 15. Test 1 Real (L) vs On-board (R) Location Measurement**

For yaw variance, two locations on the vehicle were identified by pixel and used consistently between frames to determine changes in yaw angle from one photo to the next by simple trigonometry. This is demonstrated in Figure 16. The result of this was a table of yaw angles, from which differences, and a rate of change could be calculated.

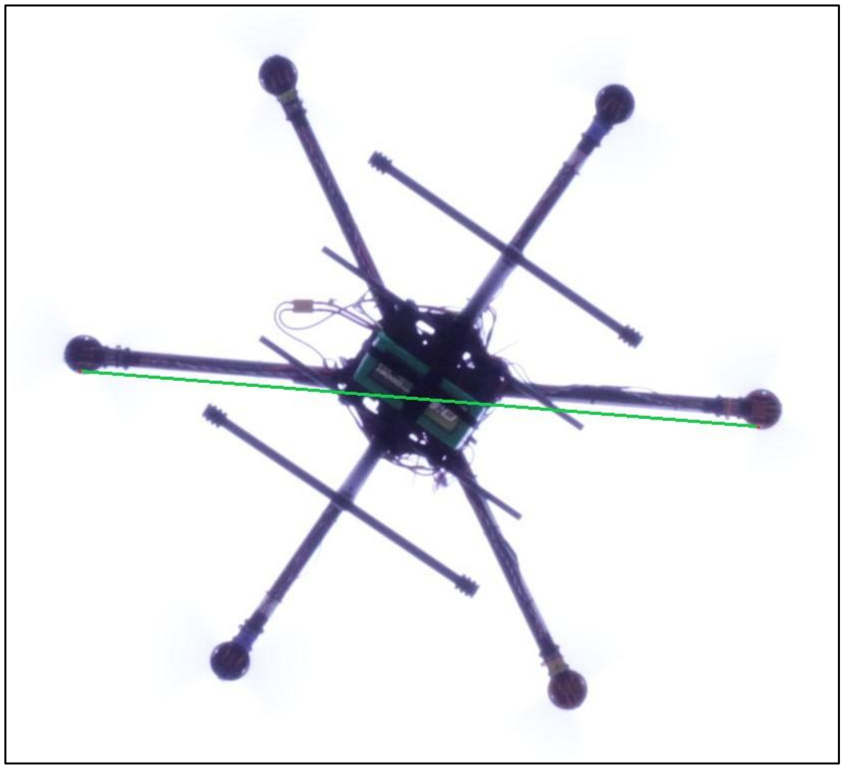
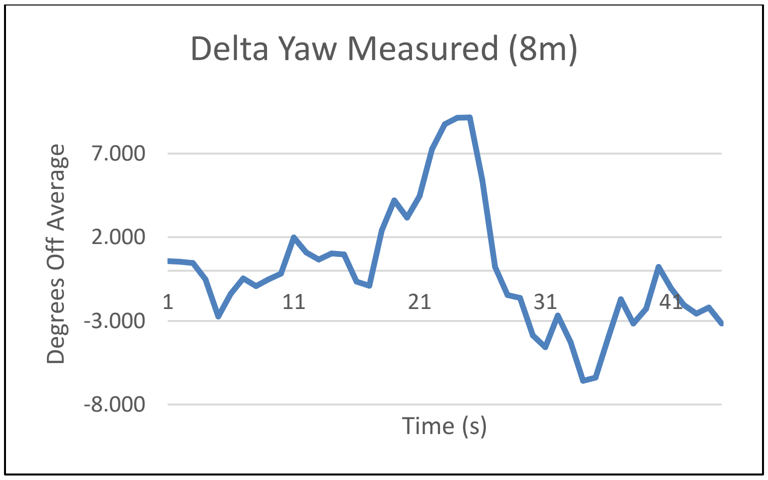
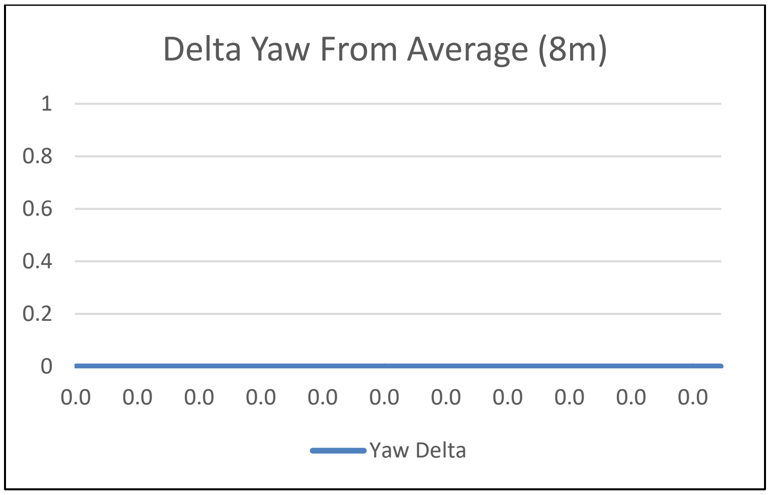


**Figure 16. Yaw Angle Measurement Example**

The results of these measurements are detailed for each test in Figure 17 and Figure 18. The overall yaw variance is remarkably limited. The yaw is largely controlled, in this autopilot, by the IMU sensors, and changes to the PID values might provide improvements to the yaw variance. The results, Figure 17, show that in both tests the vehicle remains within ±10 degrees of the initial direction. It should be noted that the difference in time between measured and onboard is related to having unsynchronized time references. The standard of deviation between the two tests ranged from 1.7° to 3.7° from the average pointing position, with an average rate of change of 0.06°/sec to 0.08°/sec.

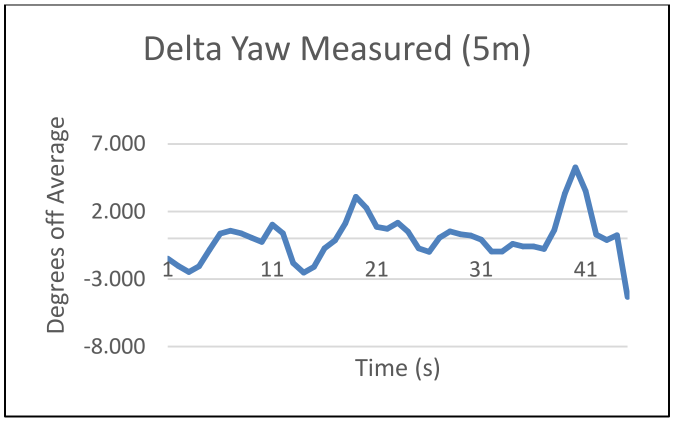


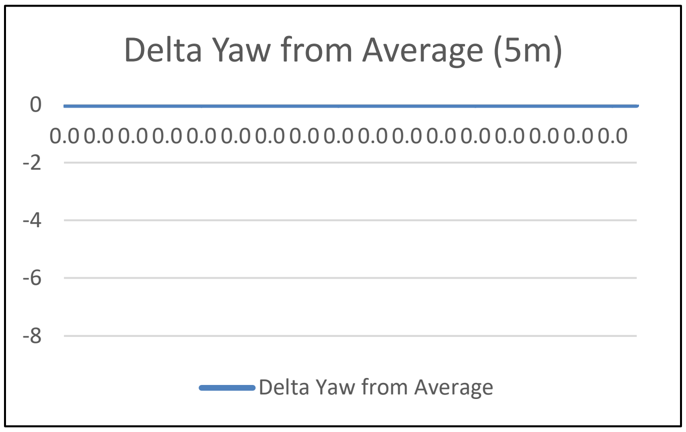


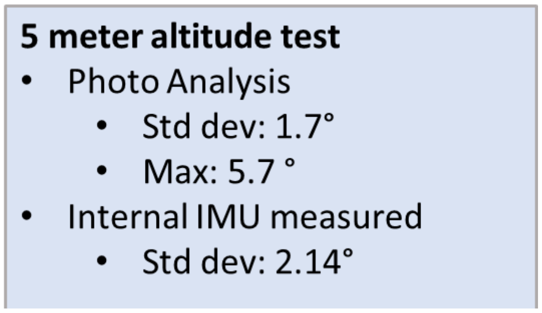
**8 meter altitude test**

- Photo Analysis
    - Std dev: 3.7°
    - Max: 9.6 °
- Internal IMU measured
    - Std dev: 2.4°

**Figure 17. Measured (Top) vs Onboard (Bottom) Yaw Variance Test 1**







**5 meter altitude test**

- Photo Analysis
  - Std dev: 1.7°
  - Max: 5.7 °
- Internal IMU measured
  - Std dev: 2.14°

**Figure 18. Measured (Top) vs Onboard (Bottom) Yaw Variance Test 2**

## 4.3 Discussion

The results of these tests tell us a great deal about the potential viability of an RTK GPS system for informing a structure navigation system. Whether a mission sensor will be capable of handling variance in distances equal to those determined in these tests would remain to be seen. That said, if a sensor can handle the following limits on the RTK GPS, reducing the environmental sensor requirements, a great deal of efficiency gains stand to be realized.

**Table 2. Sensor Parameter Limits**

| Parameter | Limits |
|---|---|
| Locational Precision | ± 40.2cm |
| Yaw Precision | ± 9.1° |
| Yaw rate | ± .08°/sec |

These limits need to be recognized as optimal conditions for the Here+ RTK system. These were found while using the system in an open field, with minimal opportunities for multi-path error, and all tests required several minutes of waiting for the RTK fix to be accomplished between the base station and the autopilot. There is a possibility that attempting to use this system near structures or in urban valleys may result in higher locating errors, or issues gaining RTK fix. All GPS based systems would encounter these potential issues, but the need for direct communication with a base station adds another communications link that needs to be maintained and is subject to interference.

Altogether, the results of this test regime are supportive of additional testing in more challenging environments. In order to better classify the suitability of this technology for localization, there are quite a few more questions to be answered. Follow on research opportunities and suggestions will be discussed in Chapter 6.

## 4.4 Summary

RTK GPS provides a powerful tool for localization, significantly improving on the accuracy of the SPP GPS signal most commonly utilized in commercial and industrial applications. This added level of accuracy comes with some additional questions, especially with regards to this specific system. Fix lag and fix loss were common problems in an environment that should have been ideal conditions for the system. In light of this, further research is required before recommendations for use in an urban environment could be given. Initial tests look promising, but there are still many questions that need attention.

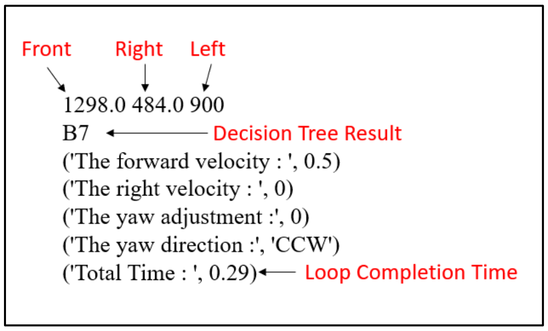
# V. Wall Following Results and Discussion

## 5.1 Chapter Overview

In this chapter, the results of the wall following portion of this thesis will be detailed. The capability of the algorithm to react appropriately to the environment, and produce reasonable velocity outputs, has been captured in a series of cart tests. These tests were built as detailed in chapter 3 and designed to produce specific outcomes. These tests are not sufficient to certify the algorithm for operations but are an acceptable starting point for limited flight tests with the appropriate revisions to the flight parameters.

## 5.2 Results

The final design of the wall following algorithm was a decision tree. The full code is recreated in Appendix B. In this figure the architecture of the decision gates for the various regions of interest (ROI) are displayed. The result is a code that is able to finish a loop every third of a second while accurately producing the reactions expected of the system in each designed scenario. A closer look at these reactions now follow. In the examples used, a print out of individual control loops will be presented, like the one in Figure 19. At the top, the measure (or resolved) distances in millimeters are shown for Front, Right, and Left ROIs respectively. The next line is the branch label from the decision tree section of the code those measurements match. The next sections are labelled, it should be noted that the “yaw adjustment” line is for specific situations like being in a corner or needing to turn around in a narrow corridor. The “yaw direction” line is the direction of yaw occurring at the defined yaw rate from the parameter file. Finally, the “total time” measure is based on a measurement of the loop from start to finish, to determine the possible input rate from the algorithm.



```
1298.0 484.0 900
B7
('The forward velocity : ', 0.5)
('The right velocity : ', 0)
('The yaw adjustment :', 0)
('The yaw direction :', 'CCW')
('Total Time : ', 0.29)
```

**Figure 19. Algorithm Output**

### 5.2.1 Distance Adjustment Tests

In the algorithm designed for this project, the vehicle is prioritizing a distance from the wall off the right side of the vehicle, in order to simulate a hard-mounted mission sensor facing perpendicular and right to the forward motion of the multi-rotor. Maintaining this consistency simplified the code but changing the algorithm to support a left facing design would be relatively trivial. The set-up is shown in Figure 20 for “Too Far” and the “Too Near” set-up are shown in Figure 21, which provided the necessary confidence that the LIDAR was taking accurate readings, and the wall surface was providing sufficient returns. The control of this distance correction was a simple linear formula of the variety described in Equation 1. For this actual test the correction factor was defined as .005 (m/mm*s), to translate the offset from an ideal distance into a translational right velocity vector at a reasonable rate (<2m/s).

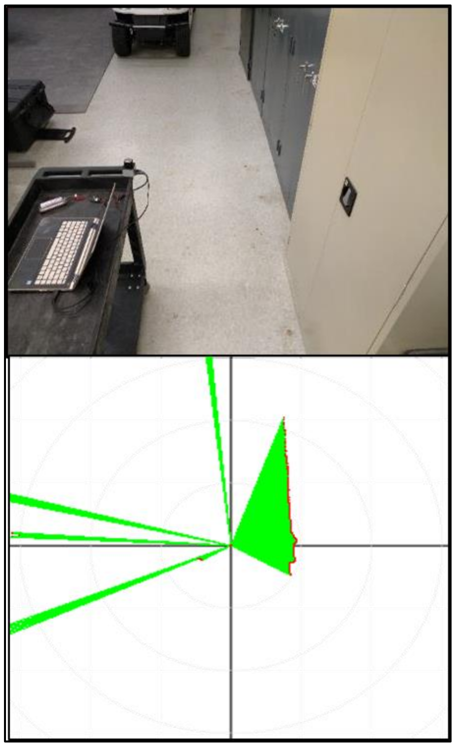


**Figure 20. “Too Far” Test Set-up and LIDAR**

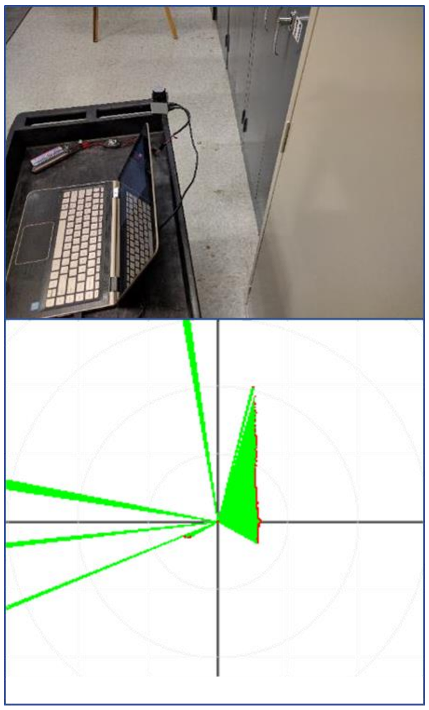
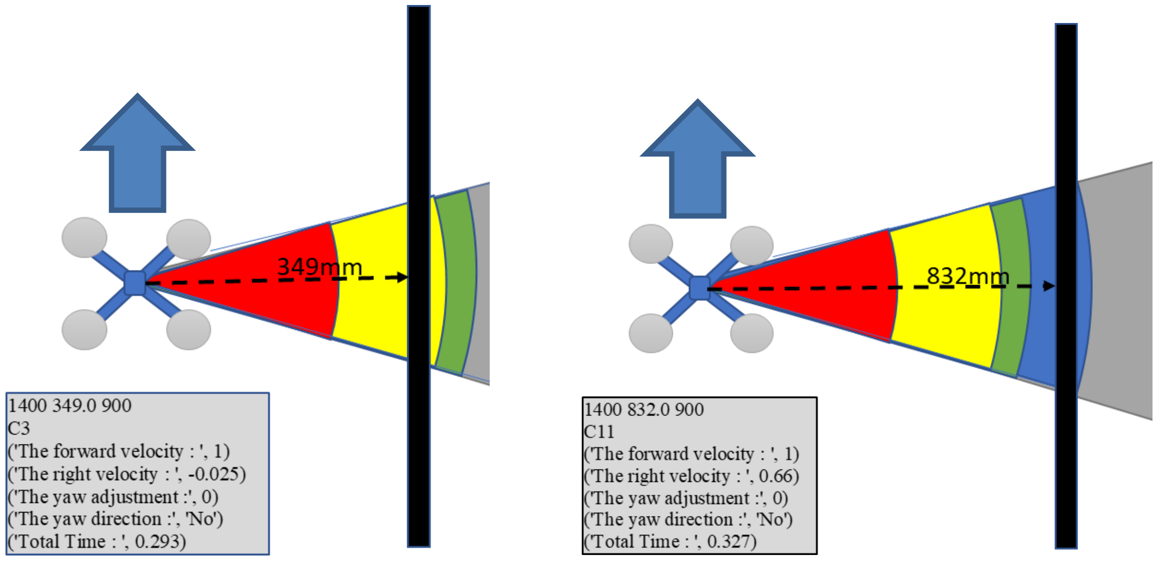


**Figure 21. “Too Near” Test Set-Up**

From these tests the code demonstrated the capability to perform a full analysis and characterization of its surroundings, determine how far away from the acceptable range it was to the right, and output a right (or left) velocity vector proportional to how far out of tolerance it is. An example of two such outputs from the control loop are captured in Figure 22, with the measurements and scenario designator (from the logic table) included for further background.



```
1400 349.0 900
C3
('The forward velocity : ', 1)
('The right velocity : ', -0.025)
('The yaw adjustment :', 0)
('The yaw direction :', 'No')
('Total Time : ', 0.293)
```

```
1400 832.0 900
C11
('The forward velocity : ', 1)
('The right velocity : ', 0.66)
('The yaw adjustment :', 0)
('The yaw direction :', 'No')
('Total Time : ', 0.327)
```

**Figure 22. “Too Near” and “Too Far” Example Output**

### 5.2.2 Corner Negotiating Tests

In these tests, the ability of the algorithm to handle both open and closed corners was evaluated. The design of the system intends that once the open or closed corner scenario is encountered the anticipated response begins. For the open corner, the vehicle continues until it is past the edge of the corner, then begins to yaw right until it is once again parallel to the wall. The closed corner scenario begins by slowing down the vehicle and begins a yaw left to align itself with the new wall at its front. If the next threshold is crossed, the vehicle stops and continues yawing left until it is aligned with the wall that was in front of it. This combination of reactions allows for the greatest potential to react correctly to a scenario, while still being controllable and predictable.

The open corner test was demonstrated on the corner below in Figure 23. The corner itself wasn't completely free of additional obstacles, but the code managed to negotiate the corner before responding to the obstacles it encountered on the far side. This gives further confidence that the system is capable of responding to a robust selection of features and scenarios. A LIDAR return snapshot was not captured for the open corner scenario, but the readings in the code reflected accurate measurements of the scenario as it played out. A sample of those returns and the algorithm reaction are captured in Figure 24.

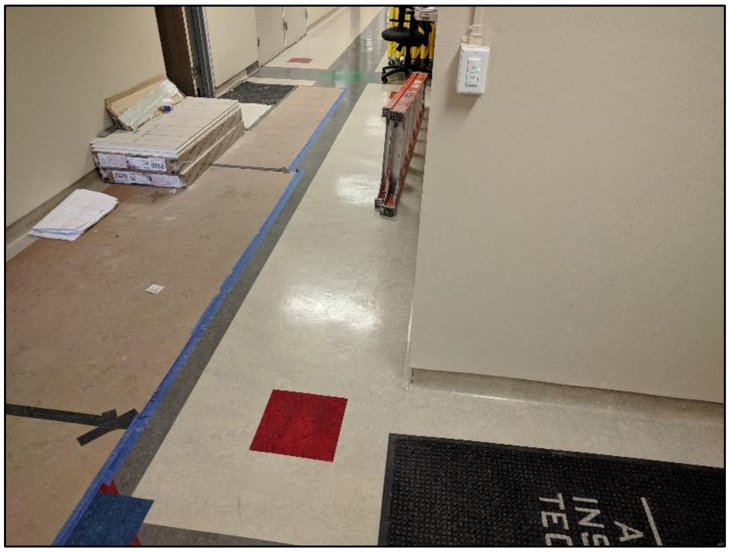
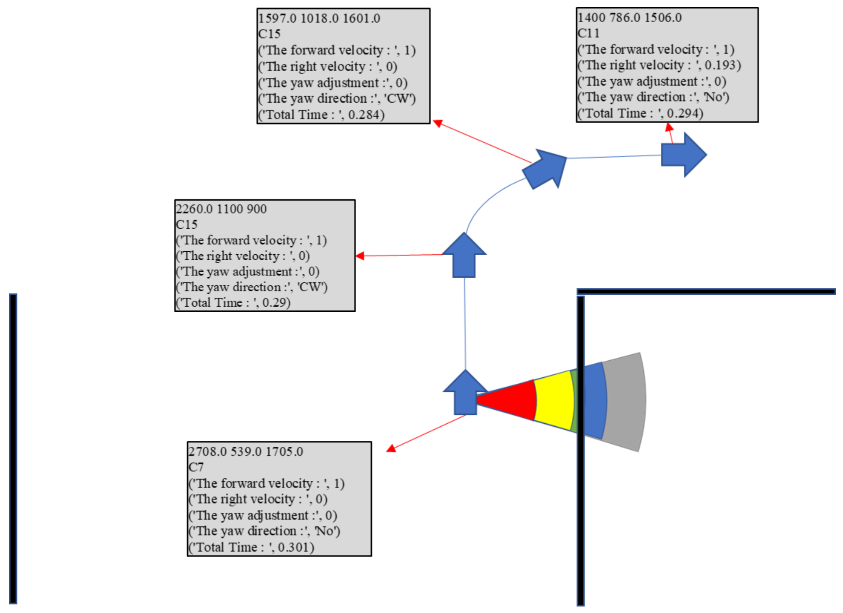


**Figure 23. Open Corner Test Location**



```
1597.0 1018.0 1601.0
C15
('The forward velocity : ', 1)
('The right velocity : ', 0)
('The yaw adjustment :', 0)
('The yaw direction :', 'CW')
('Total Time : ', 0.284)
```

```
1400 786.0 1506.0
C11
('The forward velocity : ', 1)
('The right velocity : ', 0.193)
('The yaw adjustment :', 0)
('The yaw direction :', 'No')
('Total Time : ', 0.294)
```

```
2260.0 1100 900
C15
('The forward velocity : ', 1)
('The right velocity : ', 0)
('The yaw adjustment :', 0)
('The yaw direction :', 'CW')
('Total Time : ', 0.29)
```

```
2708.0 539.0 1705.0
C7
('The forward velocity : ', 1)
('The right velocity : ', 0)
('The yaw adjustment :', 0)
('The yaw direction :', 'No')
('Total Time : ', 0.301)
```

**Figure 24. Open Corner Returns and Command Example**

The closed corner test was constructed as shown in Figure 25. In this scenario the algorithm correctly approached the wall until the vehicle crossed the first threshold, at which point it began slowing forward velocity, and introducing a yaw rate. Finally, once the vehicle was in the final threshold it stopped completely and started a 90-degree yaw to the left. All of these commands were sent and updated on a ~0.3s basis, allowing for a consistent update to the control algorithm depending on how the environment changed. A sample of the returns (Figure 26) and algorithm reactions for each step (Figure 27) in the test are shown.

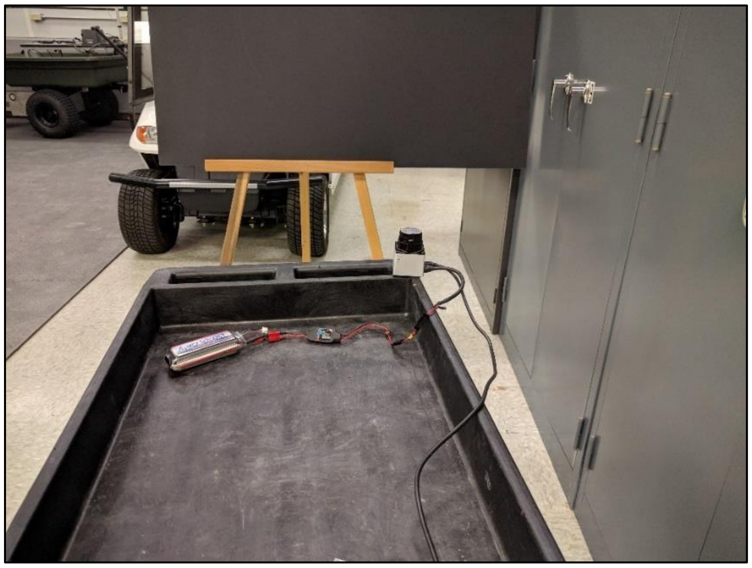


**Figure 25. Closed Corner Test Set-up**

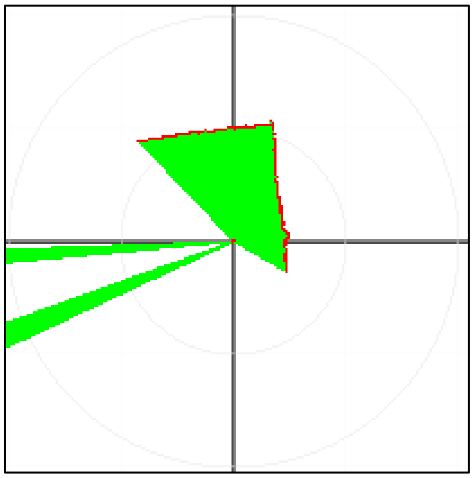
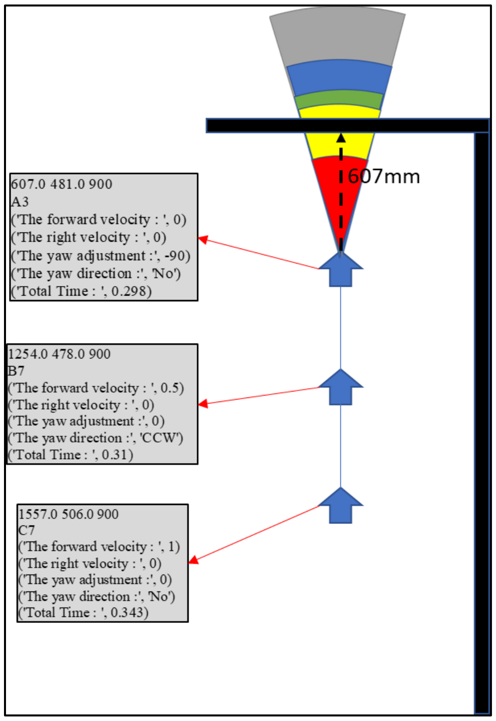


**Figure 26. Closed Corner LIDAR Vision**



```
607.0 481.0 900
A3
('The forward velocity : ', 0)
('The right velocity : ', 0)
('The yaw adjustment :', -90)
('The yaw direction :', 'No')
('Total Time : ', 0.298)
```

```
1254.0 478.0 900
B7
('The forward velocity : ', 0.5)
('The right velocity : ', 0)
('The yaw adjustment :', 0)
('The yaw direction :', 'CCW')
('Total Time : ', 0.31)
```

```
1557.0 506.0 900
C7
('The forward velocity : ', 1)
('The right velocity : ', 0)
('The yaw adjustment :', 0)
('The yaw direction :', 'No')
('Total Time : ', 0.343)
```

**Figure 27. Approaching Corner**

### 5.2.3 Impinging Walls Test

In this test the algorithm was faced with a set of walls closing in on either side as it moved forward. This scenario tested the vehicles ability to react to a closing corridor situation. It is designed to proceed at its normal velocity until the walls on both sides have passed into the minimum distance region, at which point it comes to a stop completely and yaws a full 180-degrees. The set-up of this test is shown in Figure 28 below. The LIDAR returned sufficient data (Figure 29) to ensure awareness of the incoming walls and react appropriately to the quickly reducing maneuvering room. As the vehicle crossed the necessary thresholds it went from prioritizing the mission sensor stand-off to exiting the corridor (Figure 30).

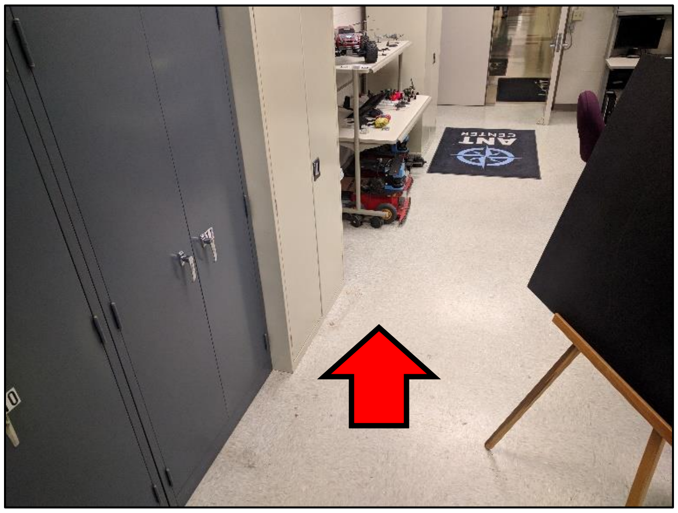


Figure 28. Impinging Walls Set-up

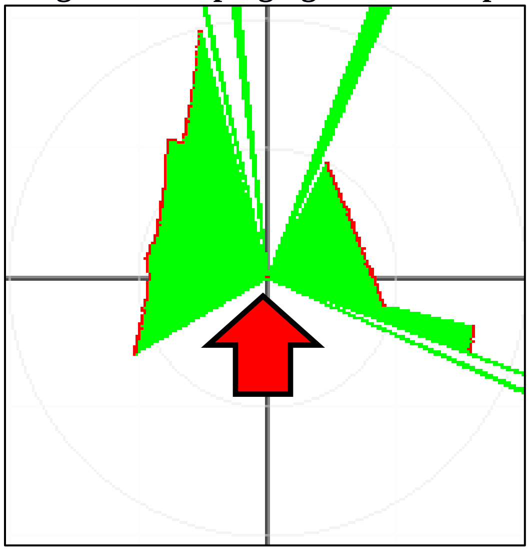
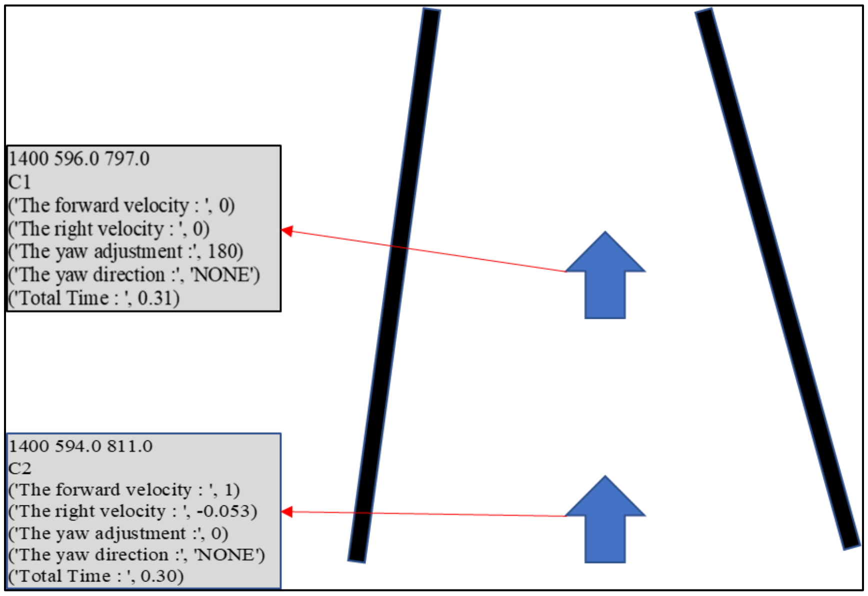


Figure 29. Impinging Walls LIDAR Returns



```
1400 596.0 797.0
C1
('The forward velocity : ', 0)
('The right velocity : ', 0)
('The yaw adjustment :', 180)
('The yaw direction :', 'NONE')
('Total Time : ', 0.31)
```

```
1400 594.0 811.0
C2
('The forward velocity : ', 1)
('The right velocity : ', -0.053)
('The yaw adjustment :', 0)
('The yaw direction :', 'NONE')
('Total Time : ', 0.30)
```

**Figure 30. Closing Corridor**

## 5.3 Discussion

Based on the results of these tests, it would be appropriate to proceed to limited flight test of the algorithm in designed scenarios. The algorithm shows sufficient robustness to handle various designed complex environment scenarios, and the necessary sampling rate to ensure dangers do not appear without being identified within the ROIs of the system. The question of sufficiency and robustness are at this stage only partially explored, as real-world testing and potential novel scenarios may be encountered and would require further testing to ensure the algorithm responses are proper and safe in such environments. Further testing will be necessary to demonstrate specific vehicle variances in yaw and position, which are closely tied to the vehicle dynamics and control schema utilized.

## 5.4 Summary

In all, these tests are a strong first step towards vehicle implementation of a wall-following algorithm. In all scenarios, the algorithm responded with the appropriate reactions at a sufficient rate to limit danger to the vehicle, but still require further and more robust on-vehicle testing going forward. The next chapter will take these results and discuss the conclusions of the research and recommendations for future work in support of this topic.

# Conclusions and Recommendations

## 6.1 Overview

In this chapter, a review of the investigative questions, and the conclusions that can be drawn regarding them based on this research will be performed. A determination of how much of the problem space was explored and what remains to be researched must be performed still. The significance of the research accomplished will be detailed with respect both the general body of related science, and the CONOPs of the sponsoring organization. Next, recommendations for action related to this specific thesis work will be proposed to either close-out investigative questions or deliver more robust conclusions to them. Finally, recommendations for future research will be detailed to support the overall CONOPs requirements which could not be met under the constraints of this thesis.

## 6.2 Conclusions of Research

Based on the tests and research performed under this thesis the following conclusions can be drawn. A wall following algorithm based on a decision tree type logic model, tempered by reaction curves that will allow the vehicle to avoid logic lock, is a potentially sufficient answer to the question of proximity flight. Especially for a multi-rotor UAV used to support mission sensor operational parameters. The question of whether RTK GPS has sufficient positional accuracy to merit further investigation as a replacement to onboard sensing systems has been answered as possibly, subject to additional research.

Reviewing the investigative questions individually, the first regarding the use of sensors to navigate along a wall has been demonstrated as feasible. The consistent output of acceptable control messages can be achieved using a decision tree algorithm and a small LIDAR system. Decision tree style wall-following is a simple, computationally cheap, and highly expandable approach to the problem of proximity UAV flight. With proper region of interest definition and zone creation, the vehicle can be expected to keep itself within operating parameters in the most typical wall following scenarios. The system has also shown itself capable of dealing with several more complex environmental scenarios. This approach has the benefit of allowing the designer to create a highly defined set of actions for the vehicle, removing uncertainty in the vehicles reactions to particular situations. This strength is also, in some ways, its weakness. The capability of the system to react to the environment is also highly constrained to the imagination and thoroughness of the designers developed responses. This leaves open the possibility of the vehicle being endangered by a novel and unanticipated environmental scenario given the lack of reactionary flexibility.

The second investigative question regarding the use of a sensor suite to hold the vehicle stationary, was combined with the fourth question regarding the use of RTK-GPS to hold the vehicle stationary. The use of on-board sensing was not accomplished, although there are opportunities to do so, but RTK-GPS positional accuracy on a multi-rotor is a very promising approach for localizing a vehicle with accuracy much greater than that of traditional GPS. The RTK-GPS holds to well within 1-meter accuracy a vehicle with its normal position hold control dynamics. The sufficiency of this accuracy will be highly dependent on the application attempting to utilize it. If 1-meter accuracy is sufficient for the mission sensor and path limitations, this would be an approach deserving of further inquiry. The concerns with this approach are those inherent to any GPS based system. The potential for intentional jamming is always a serious concern for operational systems utilizing GPS. Additionally, all GPS solutions suffer performance degradation in urban areas as a result of multi-path effects. The impact of such effects were not investigated in this thesis. Also, of note, the ability of this particular UAV sized RTK-GPS solution to achieve and maintain RTK fix was intermittent at best. Losing RTK fix during an operational action that relied on this level of GPS accuracy could undermine the mission sensor effectiveness at the very least and endanger the mission vehicle and the expensive mission equipment by drifting into obstructions in the worst-case scenario. As a result, further research will be recommended, and as of the results of this project, it cannot be recommended as a standalone localization solution.

The third investigative question relates to the precision of a wall following system utilizing on-board sensors. The application of this thesis' wall-following algorithm to an air vehicle was not accomplished, but the cart testing does provide some insight. The loop rate of ~0.3s, will allow the vehicle to make consistent updates to the system, but tuning of the max yaw rate, and right velocity factor will be necessary to adjust the system dynamics and precision.

The fifth investigative question of yaw variance for a sensor in hover was answered in the RTK-GPS testing. The combination of the location control, and IMU inputs limited the standard of deviation of the yaw to within 1.7° and 3.7° respectively, with the max departure being 10° on the 8m altitude flight. Further testing and the operating parameters of the mission sensor will determine whether this approach is sufficient for operational usage. The final investigative question of yaw variance in motion was not accomplished in this thesis, as neither RTK-GPS or the wall-following algorithm were tested on a vehicle in motion.

## 6.3 Significance of Research

The outcome of this research provides a number of useful insights with respect to the application of multi-rotor structure proximity flight. The wall-following algorithm provides a strong basis for both the expanded role of multi-rotor systems near buildings, and also the ability of those systems to support mission sensors with specific operating requirements for distance or movement pace. The RTK-GPS research provides a good starting point for understanding the limitations and opportunities available to systems requiring higher accuracy localization than is available by standard GPS solutions. A potential to minimize on board sensing requirements in GPS permissive, static obstacle scenarios has been shown to exist. Further, the overall architecture of a structure proximity flight system has been defined and partially investigated for multi-rotor UAVs, providing a roadmap for follow-on research in this topic, and for the implementation of the full CONOPs.

## 6.4 Recommendations for Action

As of the completion of this thesis, several additional issues require attention. The wall-following algorithm shows robustness in scenarios with decisive LIDAR returns and near uniform wall patterns. Going forward, additional ROIs should be included to provide the opportunity to react to deep but narrow breaks in the structure face like open doorways or similar features. A more robust means of controlling yaw between readings should be investigated, potentially one that turns the LIDAR readings into a line that the heading can be made parallel to, as opposed to the current approach which attempts to find the minimum distance return and adjusts the heading to put that return 90-degrees off the heading. The current approach leaves it in danger of getting single errant returns off of materials that have poor light reflectivity or even debris in the air. Additionally, the data gating approach used in this algorithm leaves the system susceptible to low profile obstacles that stand out significantly from their surroundings. There are a wide variety of potential actions and improvements that might be applied to the wall-following algorithm as it is currently designed that could yield gains in response robustness and safety.

Investigating the use of lower power and lighter ultrasonic sensors might be useful for dealing with some of the issues associated with poor light reflectivity surfaces. The inclusion of ultrasonic sensors either as supplements to provide more robust reading returns, or chained together for a full 360° sensing capability, might be one way of improving the vehicles safety and stability.

With respect to the RTK-GPS system, consideration should be given to testing a few of the other commercially available RTK-GPS solutions for commercial operations. It’s possible that the fix holding concerns and accuracy could stand to be greatly improved by different solutions already in existence on the market such as other u-blox M8P solutions like Drotech XL RTK or the standalone ComNav K501g L1/L2 system. The prices of any of these systems is not negligible, but other hobbyists have had good experiences with them, which might merit investigating them further.

## 6.5 Recommendations for Future Research

Moving forward from here, the remaining portions of the CONOPs would be a good place to start for follow-on research. As mentioned in the literature review, research related to localization and mapping, and path finding has a deep library of previous work but fusing those approaches to include wall-following could provide some very useful advances, especially for the sponsor. Creating a solution that allows an operator to navigate the faces of a structure safely and accurately will require more than just wall-following algorithms to be operationally relevant. The decision to use LIDAR as the sole sensing system can also be improved on. There is a distinct possibility that ultrasonic or optical flow sensing could be utilized for forward and opposite mission sensor directions, where the squareness to the face does not require pulling information from the LIDAR return angles. The research performed by AFIT students, examining photogrammetry and a unified behavior framework for path finding might be good gateways into this new fusion of algorithms for a comprehensive proximity flight solution. Overall, the next step would be an outstanding opportunity for a "full systems engineering approach" to organizing these disparate algorithms into a cohesive solution set. Inputs from the sensing suite will require control of the form and data type, and outputs will need to be deconflicted for priority and safety.

With regards to RTK-GPS, research needs to be completed with a moving system to determine localization accuracy under way. Additionally, research on RTK-GPS solutions while operating near structures to determine the effects of multi-path interference will be necessary to classify their efficacy in this application. Multi-path interference is a known performance concern for all GPS solutions, and RTK is particularly susceptible to link interferences. As such, this branch of the thesis is well positioned for follow-on efforts to characterize the actual impacts of these effects on the system.

Generally, the impacts of wind vortexes in urban canyons need to be researched to classify the potential impact on any stability algorithm, whether sensor based or utilizing GPS-RTK. As is, the algorithm designed in this thesis stands to be further tested on Rovers or in a controlled multi-rotor environment.

## 6.6 Summary

In conclusion, the results of this thesis provide a strong step forward for the application of multi-rotor flight in proximity to structures in support of a mission sensor. The wall-following algorithm is ready for a new phase of testing, and has some known blind-spots that, if addressed, would provide significant utility as a navigation aid. The ability of RTK GPS to provide localization has also been shown to have promise as either an enhancement to other approaches or even a stand-alone solution, where sub meter accuracy is required.

**Appendix A. Concept of Operations (CONOPS)**

## I. Purpose

This document describes employment scenarios whereby the thesis System outfitted with an advanced sensor suite is used to perform proximity sensing operations on various structures.

## II. Background

Operating Remotely Piloted Aircraft (RPA) in close proximity to buildings is a high-risk operation given the potential for collision as a result of wind direction change, gusts, protruding features, and many other factors. In spite of this, there are many applications which demand proximity flight to make optimal use of sensors, provide support to missions, and navigate congested RPA flight paths.

The sponsor has a requirement to operate an RPA in close proximity to a structure and be capable of both stable operations near that structure, and accurately holding position. The ability of this system to perform these tasks, ideally with as little human input as possible.

## III. Future Environment

(AF Urban Ops Vision, ISR (building interior), etc)

- Speak with AFSOC SOCOM
- "The Joint Force of 2020 will…capitalize on emerging joint operations as SOF, Cyber, robotics, and ISR as central to joint operations and leverage game changing capabilities to enhance smaller well trained and equipped force"
-

Final Report of the Maneuver and Mobility Concept Team," from CSA SSG II, available at the General Officer Management Office

(Quoted in a RAND Corp Future Urban Ops Report)

## IV. Concept Time Frame/Scope

The system being demonstrated in part by this project could be developed and deployed within five years, assuming an enduring requirement, and the continued development of the mission sensor, or equivalent, into an operationally deployable system.

The scope of this project CONOPs is the development and demonstration of a structure tracking and traversing UAS capable of carrying a mission sensor representative payload and that sensor's required systems support architecture utilizing Line-of-Sight (LOS) communications and nearby operators. While the final form of the UAS being developed is unlikely to meet the exact requirements of a final sensor payload design specification, this architecture will be readily adaptable to most variations.

## V. Military Need Statement

The sponsor has need of an UAS capable of carrying its sensor around targeted structures while maintaining flight profiles for stability, consistent speed, and holding fixed location while putting the sensor itself at minimal risk to damage or loss.

The difficulty of proximity operations is tied to the complexity of any given structure face, features, and the unpredictability of air currents around them. The ability of a human operator to correctly identify a change in the surrounding air flow, keep situational awareness of all surrounding structures and irregular features, and finally be able to correct the flight path before coming into contact with any of these is very limiting. Ideally a machine capable of maintaining some basic level of situational awareness and being programmed to make changes in its attitude and flight profile given those influences would be much better suited to react quickly and decisively to any changes in its surroundings or flight dynamics and reduce or eventually eliminate the overall situational awareness burden on the human operator.

## VI. Sequenced Actions

The use of the UAV system would follow these actions:

- Setup:
    - Operators deploy GCS and assemble UAV, install and check payload functionality, perform all pre-flight checks.
- Mission Planning
    - Includes all actions required to direct a UAS to the structure of interest with little to no prior surveying accomplished beyond simple satellite location and size data.
- Launch:
    - Includes all actions required to bring the UAV to stable flight and the initial mission waypoint
- Navigation
    - Includes the following of the pre-determined flight path towards the mission area. May include an obstacle avoidance mode for reaching mission start
- Mission Execution
    - Perform all actions required to complete mission including, either layout survey or activity monitor are selectable modes, operator selects desired sensor output.
        - Surveying Mode: UAV is guided around the exterior of the building at a controlled speed utilizing its sensor and GPS data
        - Hold Position Mode: UAV is positioned outside the structure and made to hold a single position accurately relative to the structure
    - UAV automatically avoids obstacles while tracking along structure surface
    - Operator provides mode switching

    - Mission data uplink to ground station and/or stored locally for download post mission
- Mission Extraction
    - Includes all actions and algorithm required to follow pre-determined route back to launch location, or other specified landing zone, may include an obstacle avoidance mode for reaching recovery location
    - If signal is lost, UAV automatically navigates away from building and follows ingress route in reverse to launch location or previously assigned recovery location
- Recovery
    - Includes any actions required to allow the UAV to land in its assigned landing zone
- Mission data exploitation/tear-down
    - Includes any actions required for data exploitation, and dissemination
    - Includes any actions required for GCS pack up, UAV disassembly or prep for re-launch.

## VII. Central Idea / Vision Statement

The central idea for this capability is that as military and police forces continue to operate in and around urban environments, they are frequently presented with situations where a structure either holds enemy combatants, criminals, or its layout and occupancy status are simply unknown. These scenarios are extremely dangerous for the military, police forces, and civilians caught in them. In these situations, more information leads to better informed decision making and commensurately better outcomes. This CONOPs describes the use of a UAV that carries the mission payload for investigating the characteristics of a structures interior, while operating within the mission sensors operational requirements.

## VIII. Capabilities

### • Sensing

- Can perform mission sensor capabilities
- Can locate nearby structure faces and determine distance.
- Can determine when structure features pose a threat to the RPA on its current flight path

### • Intelligent Path Building

- Can use sensed data to build flight paths that steer clear of identified features on current heading
- Can notify operators when flight path options do not meet sensor requirements for proximity or stability.

## IX. Assumptions & Risks

This CONOPS assumes that the capability gap identified herein is still present and unresolved.  It is also assumed that the sensor or sensors being developed for the sponsor will reach a stage where they are a small enough form factor and low enough power demand, that multi-rotor designs will be the best solution for moving and locating the equipment in a mission. It is assumed that Thesis UAV will be operated locally via a LOS link. It is assumed that the operation of the Thesis UAV for this demonstration will not require low probability of detection. Finally, it is assumed that intelligence regarding the structure of interest is minimal, so basic size and location data will be the only available inputs for mission planning.

The following risks were derived by our project team:

- UAS will not have the carriage capacity to maintain the mission sensor aloft for the necessary mission duration
- UAS will not have adequate control to hold position with sufficient stability
- UAS will not be able to adequately detect features of surrounding structures to avoid them
- GPS multipath effects in close proximity to buildings may confuse/confound UAS algorithm
- Proximity sensors may lack adequate range to detect dangers in time for system to avoid contact
- UAS loss of downlink data to control station
- UAS loss of control signals and inputs through LOS/segmented LOS.
- Loss of UAS and sensor due to mechanical malfunction
- Loss of UAS and sensor due to software malfunction
- Damage to structures due to UAS collision
- Injury to personnel from falling debris from UAS failure or collisions

## X. Summary

The proposed concept will provide the sponsor with an effective platform for the use of its mission sensor suite or near equivalent. By sensing nearby structures and using those features as a means of maintaining the mission sensor within its nominal operating conditions, the system will be capable of meeting the needs of the customer and provide the architecture necessary to adjust the flight characteristics of the UAS to incorporate other sensors requiring proximity flight to structures.

This project includes the demonstration of only a portion of a full CONOPs capability necessary for effective operations in proximity to a structure.

## Appendix B. Wall Following Code

```
from dronekit import connect, VehicleMode, mavutil
import time
import serial
from hokuyo.driver.hokuyo import Hokuyo
from hokuyo.tools.serial_port import SerialPort
import numpy as np
from flightparams import FlightParams


uart_port = '/dev/ttyACM0'
uart_speed = 115200

laser_serial = serial.Serial(port=uart_port, baudrate=uart_speed, timeout=0.5)
port = SerialPort(laser_serial)

# define sweep regions
frontmindeg = 0 - FlightParams.frontsweepdeg/2
frontmaxdeg = 0 + FlightParams.frontsweepdeg/2
leftmindeg = -90 - FlightParams.leftsweepdeg/2
leftmaxdeg = -90 + FlightParams.leftsweepdeg/2
rightmindeg = 90 - FlightParams.rightsweepdeg/2
rightmaxdeg = 90 + FlightParams.rightsweepdeg/2

Lidar = Hokuyo(port)

mode = 'guided' # replace with MAVlink cmd when using pixhawk

while mode == 'guided':
   start_time = time.time()
   print(Lidar.laser_on())
   scan = Lidar.get_single_scan()
   print(Lidar.laser_off())

   #print scan

   names = ['angle', 'range']
   formats = [np.dtype(np.float32), np.dtype(np.float16)]
   dtype = dict(names = names, formats=formats)
   a = np.array(list(scan.items()), dtype=dtype)

   #print(repr(a))
   def dic2np(a):
      X = np.array([[]])
```

```
        for i in range(0, len(a['angle'])):
            X = np.append(X, np.array([a['angle'][i], a['range'][i]]))

        return X.reshape([682, 2])
    #print str(dic2np(a))


    # Sort the array by return angles into each ROI
    front = dic2np(a)[np.where((dic2np(a)[:,0] > frontmindeg) & (dic2np(a)[:,0] <
frontmaxdeg))]  # define the front sub array
    left = dic2np(a)[np.where((dic2np(a)[:,0] > leftmindeg) & (dic2np(a)[:,0] <
leftmaxdeg))]  # define the left sub array
    right = dic2np(a)[np.where((dic2np(a)[:,0] > rightmindeg) & (dic2np(a)[:,0] <
rightmaxdeg))]  # define the right sub array


    frontdist = front[:, 1]
    leftdist = left[:, 1]
    rightdist = right[:, 1]


    # GATING TO REMOVE ZEROS AND OUTLIERS
    frontavg = np.average(frontdist)
    frontstd = np.std(frontdist)
    frontavgp = frontavg + frontstd
    frontavgn = frontavg - frontstd
    leftavg = np.average(leftdist)
    leftstd = np.std(leftdist)
    leftavgp = leftavg + leftstd
    leftavgn = leftavg - leftstd
    rightavg = np.average(rightdist)
    rightstd = np.std(rightdist)
    rightavgp = rightavg + rightstd
    rightavgn = rightavg - rightstd
    #print frontavg
    #print frontstd

    frontdist2 = frontdist[np.where((frontdist[0:]>frontavgn) & (frontdist[0:]<frontavgp))]
    leftdist2 = leftdist[np.where((leftdist[0:]>leftavgn) & (leftdist[0:]<leftavgp))]
    rightdist2 = rightdist[np.where((rightdist[0:]>rightavgn) & (rightdist[0:]<rightavgp))]
    #print (frontdist2)

    frontmin = np.min(frontdist2[0:])
    def getfrontmin():              #Use these function to remove further zeros and small
averages
        if frontmin == 0:
            frontmin1 = np.average(frontdist2) - np.std(frontdist2)
            if frontmin1 <= 100:
                frontmin1 = FlightParams.frontthreshold2+100
            else:
                frontmin1 = frontmin1
        else:
            frontmin1 = frontmin
        return frontmin1


    leftmin = np.min(leftdist2[0:])


    def getleftmin():
        if leftmin == 0:
            leftmin1 = np.average(leftdist2)-np.std(leftdist2)
            if leftmin1 <= 100:
                leftmin1 = FlightParams.leftthreshold2 + 100
            else:
                leftmin1 = leftmin1
        else:
            leftmin1 = leftmin
        return leftmin1


    rightmin = np.min(rightdist2[0:])


    def getrightmin():
        if rightmin == 0:
            rightmin1 = np.average(rightdist2)-np.std(rightdist2)
            if rightmin1 <= 50:
                rightmin1 = FlightParams.rightthreshold3 + 100
            else:
                rightmin1 = rightmin1
        else:
            rightmin1 = rightmin
        return rightmin1

    print getfrontmin(), getrightmin(), getleftmin()

            # MAIN DECISION TREE FUNCTION
    def getvehiclevelocitybody():
        velocity = ''
        if FlightParams.frontthresholdDC <= getfrontmin() < FlightParams.frontthreshold1:
            if FlightParams.rightthresholdDC <= getrightmin() <
FlightParams.frontthreshold1:
                if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                    velocity = 'A1'
                elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                    velocity = 'A2'
                elif FlightParams.leftthreshold2 <= getleftmin():
                    velocity = 'A3'
                else:
                    velocity = 'A4'
            elif FlightParams.rightthreshold1 <= getrightmin() <
FlightParams.rightthreshold2:
                if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                    velocity = 'A5'
                elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                    velocity = 'A6'
                elif FlightParams.leftthreshold2 <= getleftmin():
                    velocity = 'A7'
                else:
                    velocity = 'A8'
            elif FlightParams.rightthreshold2 <= getrightmin() <
FlightParams.rightthreshold3:
                if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                    velocity = 'A9'
                elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                    velocity = 'A10'
                elif FlightParams.leftthreshold2 <= getleftmin():
                    velocity = 'A11'
                else:
                    velocity = 'A12'
            elif getrightmin() >= FlightParams.rightthreshold3:
                if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                    velocity = 'A13'
                elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                    velocity = 'A14'
                elif FlightParams.leftthreshold2 <= getleftmin():
                    velocity = 'A15'
                else:
                    velocity = 'A16'
            else:
                velocity = 'A0' # Danger Close Right
        elif FlightParams.frontthreshold1 <= getfrontmin() < FlightParams.frontthreshold2:
            if FlightParams.rightthresholdDC <= getrightmin() <
FlightParams.rightthreshold1:
                if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                    velocity = 'B1'
                elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                    velocity = 'B2'
                elif FlightParams.leftthreshold2 <= getleftmin():
                    velocity = 'B3'
                else:
                    velocity = 'B4'
            elif FlightParams.rightthreshold1 <= getrightmin() <
FlightParams.rightthreshold2:
                if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                    velocity = 'B5'
                elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                    velocity = 'B6'
                elif FlightParams.leftthreshold2 <= getleftmin():
                    velocity = 'B7'
                else:
                    velocity = 'B8'
            elif FlightParams.rightthreshold2 <= getrightmin() <
FlightParams.rightthreshold3:
                if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                    velocity = 'B9'
                elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                    velocity = 'B10'
                elif FlightParams.leftthreshold2 <= getleftmin():
                    velocity = 'B11'
                else:
                    velocity = 'B12'
            elif getrightmin() >= FlightParams.rightthreshold3:
                if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                    velocity = 'B13'
                elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                    velocity = 'B14'
                elif FlightParams.leftthreshold2 <= getleftmin():
                    velocity = 'B15'
                else:
                    velocity = 'B16'
            else:
                velocity = 'B0' # Danger Close Right
        elif FlightParams.frontthreshold2 <= getfrontmin():
            if FlightParams.rightthresholdDC <= getrightmin() <
FlightParams.rightthreshold1:
                if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
```

```
                velocity = 'C1'
            elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                velocity = 'C2'
            elif FlightParams.leftthreshold2 <= getleftmin():
                velocity = 'C3'
            else:
                velocity = 'C4'
        elif FlightParams.rightthreshold1 <= getrightmin() <
FlightParams.rightthreshold2:
            if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                velocity = 'C5'
            elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                velocity = 'C6'
            elif FlightParams.leftthreshold2 <= getleftmin():
                velocity = 'C7'
            else:
                velocity = 'C8'
        elif FlightParams.rightthreshold2 <= getrightmin() <
FlightParams.rightthreshold3:
            if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                velocity = 'C9'
            elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                velocity = 'C10'
            elif FlightParams.leftthreshold2 <= getleftmin():
                velocity = 'C11'
            else:
                velocity = 'C12'
        elif getrightmin() >= FlightParams.rightthreshold3:
            if FlightParams.leftthresholdDC <= getleftmin() < FlightParams.leftthreshold1:
                velocity = 'C13'
            elif FlightParams.leftthreshold1 <= getleftmin() < FlightParams.leftthreshold2:
                velocity = 'C14'
            elif FlightParams.leftthreshold2 <= getleftmin():
                velocity = 'C15'
            else:
                velocity = 'C16'
        else:
            velocity = 'C0' # Danger Close Right
    else:
        velocity = 'D0'

    return velocity

  print getvehiclevelocitybody()
```

```
        # Minor Yaw Adjustment Needs to be Optimized and Debugged, use the “Front,
Left, and       #Right” Array?
  #yaw adjustment
  # def condition_yaw(heading, relative=True):
  #     yawdelta = 180 - np.index(rightmin)
  #     if yawdelta:
  #         yawadj = abs(yawdelta) #number of degrees to adjust
  #     if yawdelta < 0:
  #         yawdirection = 1 # controls yaw direction, 1 is cw, -1 is ccw
  #     else
  #         yawdirection = -1
  #     if relative:
  #         is_relative = 1  # yaw relative to direction of travel
  #     else:
  #         is_relative = 0  # yaw is an absolute angle
  #     # create the CONDITION_YAW command using command_long_encode()
  #     yawmsg = vehicle.message_factory.command_long_encode(
  #         0, 0,  # target system, target component
  #         mavutil.mavlink.MAV_CMD_CONDITION_YAW,  # command
  #         0,  # confirmation
  #         yawadj,  # param 1, yaw in degrees
  #         5,  # param 2, yaw speed deg/s
  #         yawdirection,  # param 3, direction -1 ccw, 1 cw
  #         is_relative,  # param 4, relative offset 1, absolute angle 0
  #         0, 0, 0)  # param 5 ~ 7 not used
  #     # send command to vehicle
  #     vehicle.send_mavlink(yawmsg)


  def get_right_velocity():
      right_vel = ''
      if getvehiclevelocitybody() in ['A4', 'B4', 'A8', 'B8', 'C8']:
          right_vel = (rightmin -
FlightParams.rightthreshold2)*FlightParams.Rightvelfactor # Too Far from right
      elif getvehiclevelocitybody() in ['C3', 'C2', 'B2', 'A12', 'B12', 'C12', 'B9', 'B10', 'C9',
'C10', 'C11', 'A16', 'B16', 'C13', 'C16']:
          right_vel = (rightmin -
FlightParams.rightthreshold1)*FlightParams.Rightvelfactor # Too Close to right
      elif getvehiclevelocitybody() in ['A0', 'B0', 'C0']: # Danger Close Right
          right_vel = -0.5
      else:
          right_vel = 0
      return right_vel
```

```
    def get_forward_velocity():
        forward_vel = ''
        if getvehiclevelocitybody() in ['A1', 'A2', 'A3', 'B1', 'C1', 'C4', 'A5', 'A6', 'A7', 'B5',
'C5', 'A9', 'A10', 'A11', 'A13', 'A14', 'A15', 'B4', 'B8', 'C8', 'B12', 'C12', 'A15', 'A16', 'B16',
'C13', 'C16',]:
            forward_vel = 0
        elif getvehiclevelocitybody() in ['B2', 'B3', 'B6', 'B7', 'B11', 'B13', 'B14', 'B15', 'B9',
'B10',]:
            forward_vel = 0.5 * FlightParams.velocitymax
        elif getvehiclevelocitybody() in ['A4', 'A8', 'A12', 'D0']: # Danger Close Front
            forward_vel = -0.5
        else:
            forward_vel = FlightParams.velocitymax
        return forward_vel


    def get_yaw_adjustment():
        yaw_adj = ''
        if getvehiclevelocitybody() in ['A11', 'A15', 'A3', 'A7']:
            yaw_adj = -90
        elif getvehiclevelocitybody() in ['A13', 'A14']:
            yaw_adj = 90
        elif getvehiclevelocitybody() in ['A1', 'A10', 'A2', 'A5', 'A6', 'A9', 'B1', 'B5', 'C1',
'C4', 'C5']:
            yaw_adj = 180
        else:
            yaw_adj = 0
        return yaw_adj

    def get_yaw_direction():
        yaw_direction = '' # controls yaw direction, 1 is cw, -1 is ccw
        if getvehiclevelocitybody() in ['B11', 'B3', 'B7']:
            yaw_direction = 'CCW'
        elif getvehiclevelocitybody() in ['B13', 'B14', 'B15', 'C14', 'C15']:
            yaw_direction = 'CW'
        else:
            yaw_direction = 'No'
        return yaw_direction




    Total_time = time.time() - start_time

    print("The forward velocity : ", get_forward_velocity())
    print("The right velocity : ", get_right_velocity())
    print('The yaw adjustment :', get_yaw_adjustment())
    print('The yaw direction :', get_yaw_direction())
    print("Total Time : ", Total_time)
```

# Appendix C. Logic Matrix

| Front ROI | Right ROI | Left ROI | Scenario | Reaction | Output | Forward | Right | Yaw |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | End of tight corridor | Stop. Turn 180. Forward 2 sec @ half speed | A1 | 0 | 0 | 180 |
| 1 | 3 | 2 | End of close corridor | Stop. Turn 180 | A10 | 0 | 0 | 180 |
| 1 | 3 | 3 | At corner | Stop. Turn left 90 | A11 | 0 | 0 | -90 |
| 1 | 3 | DC | Dangerously close on left, close corridor | Stop. Crab right and backwards | A12 | -0.5 | 0.5 | 0 |
| 1 | 4 | 1 | Tight corridor, end, open right | Stop. Turn right 90 | A13 | 0 | 0 | 90 |
| 1 | 4 | 2 | Close corridor end, open right | Stop. Turn right 90 | A14 | 0 | 0 | 90 |
| 1 | 4 | 3 | Obstacle straight ahead or end of wall w/ break | Stop. Turn left 90 | A15 | 0 | 0 | -90 |
| 1 | 4 | DC | Dangerously close on left, open right | Stop. Crab right | A16 | 0 | 0.5 | 0 |
| 1 | 1 | 2 | End of close corridor | Stop. Turn 180. Forward 2 sec @ half speed | A2 | 0 | 0 | 180 |
| 1 | 1 | 3 | At corner | Stop. Turn left 90 | A3 | 0 | 0 | -90 |
| 1 | 1 | DC | Dangerously close on left, tight corridor | Stop. Crab right and backwards | A4 | -0.5 | 0.5 | 0 |
| 1 | 2 | 1 | End of close corridor | Stop. Turn 180 | A5 | 0 | 0 | 180 |
| 1 | 2 | 2 | End of corridor | Stop. Turn 180 | A6 | 0 | 0 | 180 |
| 1 | 2 | 3 | At corner | Stop. Turn left 90 | A7 | 0 | 0 | -90 |
| 1 | 2 | DC | Dangerously close on left, close corridor | Stop. Crab right and backwards | A8 | -0.5 | 0.5 | 0 |
| 1 | 3 | 1 | End of close corridor | Stop. Turn 180. Forward 2 sec @ half speed | A9 | 0 | 0 | 180 |
| 2 | 1 | 1 | Approaching end of tight corridor | Stop. Turn 180 | B1 | 0 | 0 | 180 |
| 2 | 3 | 2 | Approaching end of corridor | Half forward speed, crab right | B10 | 0.5 | 0.5 | 0 |
| 2 | 3 | 3 | Approaching corner | Half forward speed, yaw left @ designated rate | B11 | 0.5 | 0 | Ccw @ rate |
| 2 | 3 | DC | Dangerously close on left, wide corridor | Stop. Crab right | B12 | 0 | 0.5 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 1 | Approaching corner | Half Speed. Yaw Right @ rate | B13 | 0.5 | 0 | cw @rate |
| 2 | 4 | 2 | Approaching corner | Half Speed. Yaw Right @ rate | B14 | 0.5 | 0 | cw @rate |
| 2 | 4 | 3 | Approaching corner | Half Speed. Yaw Right @ rate | B15 | 0.5 | 0 | cw @rate |
| 2 | 4 | DC | Approaching corner, DANGER CLOSE LEFT | Stop. Crab Right | B16 | 0 | 0.5 | 0 |
| 2 | 1 | 2 | Approaching end of close corridor | Half forward speed, crab left | B2 | 0.5 | -0.5 | 0 |
| 2 | 1 | 3 | Approaching corner | Half forward speed, yaw left @ designated speed | B3 | 0.5 | 0 | Ccw @ rate |
| 2 | 1 | DC | Dangerously close on left, close corridor | Stop. Crab right | B4 | 0 | 0.5 | 0 |
| 2 | 2 | 1 | Approaching end of close corridor | Stop. Turn 180 | B5 | 0 | 0 | 180 |
| 2 | 2 | 2 | Approaching end of corridor | Half forward speed. | B6 | 0.5 | 0 | 0 |
| 2 | 2 | 3 | Approaching corner | Half forward speed, yaw left @ designated rate | B7 | 0.5 | 0 | Ccw @ rate |
| 2 | 2 | DC | Dangerously close on left, close corridor | Stop. Crab right | B8 | 0 | 0.5 | 0 |
| 2 | 3 | 1 | Approaching end of close corridor | Half forward speed, crab right | B9 | 0.5 | 0.5 | 0 |
| 3 | 1 | 1 | Tight Corridor | Stop. Turn 180. Forward 2 sec @ half speed | C1 | 0 | 0 | 180 |
| 3 | 3 | 2 | Corridor, too far from wall | Forward, crab right | C10 | 1 | 0.5 | 0 |
| 3 | 3 | 3 | Wall Face on Right, too far | Forward, crab right | C11 | 1 | 0.5 | 0 |
| 3 | 3 | DC | Danger Close left | Stop. Crab Right | C12 | 0 | 0.5 | 0 |
| 3 | 4 | 1 | Close Obstacle Left | Stop. Crab Right | C13 | 0 | 0.5 | 0 |
| 3 | 4 | 2 | No Wall on Right | Forward. Yaw right @ rate | C14 | 1 | 0 | cw @rate |
| 3 | 4 | 3 | No wall on right | Forward. Yaw right @ rate | C15 | 1 | 0 | cw @rate |
| 3 | 4 | DC | Danger Close left | Stop. Crab Right | C16 | 0 | 0.5 | 0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | 1 | 2 | Close corridor | Forward, crab left | C2 | 1 | -0.5 | 0 |
| 3 | 1 | 3 | Open wall | Forward, crab left | C3 | 1 | -0.5 | 0 |
| 3 | 1 | DC | Dangerously close on left, close corridor | Stop. Turn 180. Forward 2 sec @ half speed | C4 | 0 | 0 | 180 |
| 3 | 2 | 1 | Close corridor | Stop. Turn 180. Forward 2 sec @ half speed | C5 | 0 | 0 | 180 |
| 3 | 2 | 2 | Close corridor | Forward | C6 | 1 | 0 | 0 |
| 3 | 2 | 3 | Along Wall | Forward | C7 | 1 | 0 | 0 |
| 3 | 2 | DC | Danger Close left | Stop. Crab Right | C8 | 0 | 0.5 | 0 |
| 3 | 3 | 1 | Corridor, too far from wall, close left | Half forward speed, crab right | C9 | 0.5 | 0.5 | 0 |

## Bibliography

[1] Anonymous, “Tactical UAVs: Defining the Missions,” *Mil. Technol.*, vol. 30, no. 7, pp. 91-92-99, 2006.

[2] R. Dietterle, “The Future Combat Systems (FCS) overview,” in *Proceedings - IEEE Military Communications Conference MILCOM*, 2005, vol. 2005.

[3] Office of the Under Secretary of Defense for Aquisition Technology Logistics, “The Role of Autonomy in DoD Systems,” *DoD Def. Sci. Board*, no. July, p. 125, 2012.

[4] A. Beyeler, J. C. Zufferey, and D. Floreano, “Vision-based control of near-obstacle flight,” in *Autonomous Robots*, 2009, vol. 27, no. 3, pp. 201–219.

[5] S. Lange, N. Sunderhauf, and P. Protzel, “A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments,” *Proc. Int. Conf. Adv. Robot.*, pp. 1–6, 2009.

[6] M. R. Endsley, “Situation Awareness In Aviation Systems,” in *Handbook of Aviation Human Factors*, 1999, pp. 257–276.

[7] B. Bury, “Proximity sensing for robots,” in *Robot Sensors, IEEE Colloquium on*, 1991, p. 3/1-318.

[8] P. Van Turennout, G. Honderd, and L. J. Van Schelven, “Wall-following control of a mobile robot,” *Proc IEEE International Conference on Robotics and Automation ICRA1992*, no. May. pp. 280–285, 1992.

[9] A. J. Davison and N. Kita, *3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain*, vol. 1, no. C. 2001, p. I-

384-I-391.

[10] D. W. Yoo, D. Y. Won, and M. J. Tahk, "Optical flow based collision avoidance of multi-rotor UAVs in urban environments," *Int. J. Aeronaut. Sp. Sci.*, vol. 12, no. 3, pp. 252–259, 2011.

[11] K. Takagi, K. Morikawa, T. Ogawa, and M. Saburi, "Road Environment Recognition Using On-vehicle LIDAR," *2006 IEEE Intell. Veh. Symp.*, pp. 120–125, 2006.

[12] Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus, "Synthetic 2D LIDAR for precise vehicle localization in 3D urban environment," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 1554–1559.

[13] S. Ramasamy, R. Sabatini, A. Gardi, and J. Liu, "LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid," *Aerosp. Sci. Technol.*, vol. 55, pp. 344–358, 2016.

[14] R. a Dain, "Developing Mobile Robot Wall-Following Algorithms Using Genetic Programming," *Appl. Intell.*, vol. 8, no. 5, pp. 33–41, 1998.

[15] R. Carelli and E. O. Freire, "Corridor navigation and wall-following stable control for sonar-based mobile robots," *Rob. Auton. Syst.*, vol. 45, no. 3–4, pp. 235–247, 2003.

[16] A. Nemati, M. Sarim, M. Hashemi, and M. Kumar, "Autonomous Wall-Following Based Navigation of Unmanned Aerial Vehicles in Indoor Environments," in *AIAA Infotech @ Aerospace*, 2015, no. JANUARY, p. 8.

[17] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for

autonomous indoor flying," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 2878–2883.

[18] J. Weingarten and R. Siegwart, "EKF-based 3D SLAM for structured environment reconstruction," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2005, pp. 2089–2094.

[19] E. Eade and T. Drummond, "Scalable monocular SLAM," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, vol. 1, pp. 469–476.

[20] J. Nikolic *et al.*, "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2014, pp. 431–437.

[21] B. Williams and I. Reid, "On combining visual SLAM and visual odometry," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, pp. 3494–3500.

[22] L. Yang, J. Qi, J. Xiao, and X. Yong, "A literature review of UAV 3D path planning," in *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, 2015, vol. 2015–March, no. March, pp. 2376–2381.

[23] J. Valente, J. Del Cerro, A. Barrientos, and D. Sanz, "Aerial coverage optimization in precision agriculture management: A musical harmony inspired approach," *Comput. Electron. Agric.*, vol. 99, pp. 153–159, 2013.

[24] F. Yan, Y.-S. Liu, and J.-Z. Xiao, "Path Planning in Complex 3D Environments Using a Probabilistic Roadmap Method," *Int. J. Autom. Comput.*, vol. 10, no. 6, pp. 525–533, 2013.

[25] E. Masehian and M. R. Amin-Naseri, “A Voronoi Diagram–Visibility Graph–Potential Field Compound Algorithm for Robot Path Planning.”

[26] Y. Feng and J. Wang, “GPS RTK Performance Characteristics and Analysis,” *J. Glob. Position. Syst.*, vol. 7, no. 1, pp. 1–8, 2008.

[27] I. L. Turner, M. D. Harley, and C. D. Drummond, “UAVs for coastal surveying,” *Coast. Eng.*, vol. 114, pp. 19–24, 2016.

[28] H. Xiang and L. Tian, “Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (UAV),” *Biosyst. Eng.*, vol. 108, no. 2, pp. 174–190, 2011.

[29] D. Nister, O. Naroditsky, and J. Bergen, “Visual odometry,” *Comput. Vis. Pattern Recognition, 2004. CVPR 2004. Proc. 2004 IEEE Comput. Soc. Conf.*, vol. 1, p. I-652-I-659 Vol.1, 2004.

[30] N. Gageik, T. Müller, and S. Montenegro, “Obstacle Detection and Collision Avoidance Using Ultrasonic Distance Sensors for an Autonomous Quadrocopter,” *Proc. UAVveek Work. Contrib.*, 2012.

[31] N. Sariff and N. Buniyamin, “An overview of autonomous mobile robot path planning algorithms,” in *Research and Development, 2006. SCOReD 2006. 4th Student Conference on*, 2006, pp. 183–188.

[32] R. (USAF) McClanahan, “IMPROVING UNMANNED AERIAL VEHICLE FORMATION FLIGHT AND SWARM COHESION BY USING COMMERCIAL OFF THE SHELF SONAR,” Air Force Institute of Technology, 2017.

[33] W. J. H. T. Center, “Global Positioning System (GPS) Standard Positioning

Service (SPS) Performance Analysis Report,"  March, pp. 1–61, 2017.

| REPORT DOCUMENTATION PAGE | | | *Form Approved* *OMB No. 074-0188* |
|---|---|---|---|
| | | | |
| **1. REPORT DATE** *(DD-MM-YYYY)* 22-03-2018 | **2. REPORT TYPE** Master's Thesis | | **3. DATES COVERED** *(From – To)* March 2017 – March 2018 |
| **TITLE AND SUBTITLE** Stabilized RPA Flight in Building Proximity Operations | | | **5a. CONTRACT NUMBER** |
| | | | **5b. GRANT NUMBER** |
| | | | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Kaniut, Michael M., Captain, USAF | | | **5d. PROJECT NUMBER** |
| | | | **5e. TASK NUMBER** |
| | | | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)** Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/ENY) 2950 Hobson Way, Building 640 WPAFB OH 45433-8865 | | | **8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GAE/ENY/10-Mxx |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** Undisclosed Sponsor | | | **10. SPONSOR/MONITOR'S ACRONYM(S)** |
| | | | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |
| **12. DISTRIBUTION/AVAILABILITY STATEMENT** APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. | | | |
| **13. SUPPLEMENTARY NOTES** | | | |
| **14. ABSTRACT** The thesis seeks a solution to the requirement for a highly reliable and capable Unmanned Air Vehicle (UAV) to support a wide array of missions and applications that require close proximity flight to structures. The scope of the project includes the drafting of a concept of operations (CONOPs) describing how the mission requirements might be met using the sensor, operators, and air vehicle described in this thesis. The demonstration of the wall-following section of that CONOPs is performed by cart testing a custom algorithm and evaluating its ability to react to its environment. Finally, a flight test was performed to characterize the capabilities of an RTK-GPS system to stably hold a UAV in a single position, and minimize vehicle yaw, as a potential means of minimizing environmental sensing requirements in GPS permissive environments. The results for RTK-GPS were, position hold standard of deviation 8.0 x 10.1cm at a 5m flight altitude, and 17cm x 12.7cm at 8m flight altitude. Yaw variation results were a standard of deviation of 1.7° at 5m and 3.7° at 8m. The LIDAR wall-following tests proved the feasibility of using a decision tree style coding approach to proximity flight near a structure, but still has some changes that should be considered before being used operationally. | | | |
| **15. SUBJECT TERMS** Fill in | | | |

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| | | | UU | 94 | David Jacques, Lt Col (Ret), USAF ADVISOR |
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | 19b. TELEPHONE NUMBER *(Include area code)* |
| U | U | U | | | (937) 255-3355, ext 3329 (david.jacques@afit.edu) |
| | | | | | **Standard Form 298 (Rev. 8-98)** Prescribed by ANSI Std. Z39-18 |