3-2021

Automated Find Fix and Track with a Medium Altitude Long Endurance Remotely Piloted Aircraft

Aubrey L. Olson

Automated Find Fix and Track with a Medium Altitude Long Endurance Remotely Piloted Aircraft

THESIS

Aubrey L. Olson, Capt, USAF

AFIT-ENG-MS-21-M-069

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-21-M-069

AUTOMATED FIND FIX AND TRACK

THESIS

Presented to the Faculty

Department of Electrical Engineering

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 Electrical Engineering

Aubrey L. Olson, B.S. Computer Science, M.S. Computational Sciences and

Robotics, M.S. Flight Test Engineering

Capt, USAF

March 3, 2021

AFIT-ENG-MS-21-M-069

AUTOMATED FIND FIX AND TRACK

THESIS

Aubrey L. Olson, B.S. Computer Science, M.S. Computational Sciences and Robotics, M.S. Flight Test Engineering
Capt, USAF

Committee Membership:

Dr. Brett Borghetti, PhD
Chair

Dr. Scott Nykl, PhD
Member

Dr. Gilbert Peterson, PhD
Member

Abstract

A ubiquitous mission set of the USAF is prosecuting military targets in civilian environments from aerial platforms in a high collateral environment. The Joint Publication 3-60, which outlines tactical doctrine, defines this as the Find, Fix, Track, Target, Engage, Assess[18] (F2T2EA) "kill chain". The first 3 phases (F2T) represent Intelligence, Surveillance, and Reconnaissance (ISR) and must be completed before engaging a target. One asset that has become prevalent in the mission set is the MQ-9 "Reaper", a remotely piloted aircraft (RPA).

A limitation in RPA ISR operations is loss of target track if the command link is severed. For an RPA to effectively execute the ISR mission without a command link, it needs the capability to F2T targets autonomously. Automated Find Fix and Track (AFFTRAC) was developed to help solve this problem. While the component algorithms were not novel, the creation of and integration into a proof of concept tactical autopilot was. Monocular stereo vision was used to process sequential images acquired during orbit to produce a partial structural point cloud of the original structure. This partial structural point cloud was then exploited to generate a holding area density for the aircraft to stay within. A simple greedy algorithm exploited this holding area density to produce aircraft turn commands to approximate tactical ISR holding. The result was that imagery from existing MQ-9 sensors was used to provide command guidance to autonomously to maintain line of sight to a target.

Generation of holding area densities was scored by a weighted classification accuracy. 55% in simulation was sufficient for correct holding. The insufficient score of 38% in flight test was likely due to a rotation error in the partial structural point clouds. The turn commands generated were 94% correct. However, sub-optimal commands did not direct the aircraft to intercept the middle of the holding area density producing oscillatory flight paths. AFFTRAC stayed within the holding area 87% of the time in simulation but only 45% in flight test due to higher airspeed and lower command latency which worsened the oscillatory flight paths. AFFTRAC maintained line of sight to the target 96% of the time in simulation and 100% in flight test. This was superior to the 82% of a constant 360-degree hold and 96% of an MQ-9 Pilot. Overall, AFFTRAC is a promising initial framework for a tactical autopilot, but additional development is needed to mature component algorithms.

Acknowledgements

I would like to thank several people that made AFFTRAC possible. First, I need to thank my wonderful wife. She is the light of my life and gives me the thing in life that everyone is looking for: true love. I am truly blessed to have been given such a gift. From our love have come two wonderful sons and a daughter. Seeing their happiness and wonderment from experiencing the world for what it is reminds me why I choose to pursue what I am passionate about even when it is difficult. Family is truly what matters most in life.

Next, I would like to thank my primary advisor, Dr. Brett Borghetti. His willingness to let me explore new territory on my own combined with timely, critical, and uncompromising feedback was critical to the development of AFFTRAC and my development as an Air Force officer. Similarly, the effort of the AFFTRAC flight test team (Capt. Samual "Bobsled" Browne, Capt. Mark "Buzz" Busby, Capt. Adam "Hawk" Fuhrmann, Capt. Eddie "Van" Hilburn, and Capt. Joseph "Simba" Schwemmer) elevated AFFTRAC to a much higher level in terms of development and execution of the flight test. I will remember those times for the rest of my life.

I would like to thank the rest of my thesis committee: Dr. Scott Nykl was instrumental as AFFTRAC ran on his AftrBurner engine. His classes made me a massively better C++ programmer. Dr Peterson provided the best classical artificial background I have seen in my academic career. I would also like to thank people at TPS: Mr. Thomas Hill for taking a chance on my project and helping me get C-12J flight time at Holloman, Lilly Aguilar for managing all the administrative details, Lt. Col Hans "Money" Buckwalter and Lt. Col Patrick "Krown" Killingsworth for sharing their flight test experience during execution, and all of the staff at TPS.

Finally, I would like to thank people at the 49th OG: Capt. Elliot "FIIG" Mcnellis and Maj. Daniel "BRAKR" McCready for laying the ground work to get MQ-9 flight time. And most of all, I would like to thank Maj. Michael Byrnes for ensuring that this flight test happened. It is not an understatement to say that he is the main reason that AFFTRAC was able to be flight tested at Holloman AFB.

Aubrey L. Olson

Table of Contents

Page

Page

Page

List of Figures

Figure Page

Figure Page

List of Tables

AUTOMATED FIND FIX AND TRACK

I. Introduction

1.1 Motivation

The current Global War On Terror (GWOT) is starkly different than past conflicts. While the threat of a conventional war still looms in the background, in the current modality of war the lines of battle are not clear. It is not obvious who the combatants actually are without comprehensive understanding of the situation on the ground. Thus, the need for precision is higher than ever. Countless hours of Intelligence, Surveillance, and Reconnaissance (ISR) must be accomplished to find a target and correctly identify it. Still, more ISR must be conducted to ensure that there will be no collateral damage in prosecution. If the goal is to strike the target, it must occur correctly the first time as there may not be another chance. The MQ-9 "Reaper" has silently become the workhorse for the United States Air Force (USAF) in this "hunter-killer" mission. This Remotely Piloted Aircraft (RPA) allows human operators to do this more effectively than ever before.

Aircraft such as the MQ-9 are perfect for this role because of their long loiter time, high-quality sensors, and unique ability to integrate into a digital intelligence process directly. This last ability is due solely to the fact that all sensor and telemetry data has been turned into a digital data stream for the purpose of remote control. This data can be sent anywhere around the world with no modification and consumed by existing intelligence apparatus. However, this remote control link is vulnerable to being severed either through hardware fault, software fault, or through adversarial

means. Thus, an RPA loses its ability to collect ISR as described above when their digital data-link is severed unless they have some sort of tactical autopilot.

Additionally, a tactical autopilot could be used even with a functional remote control link. It could be used as a work-load reduction tool allowing the aircraft crew to focus on building situational awareness (SA) or on passing intelligence. If fully mature, a tactical autopilot could even allow operators to control multiple Unmanned Aerial Vehicles (UAVs) in parallel. This thesis seeks to demonstrate that Automated Find Fix and Track in existing RPAs is possible. It seeks to create a tactical autopilot that could be used in what has become the USAF's biggest single operational time commitment.

1.2 Problem Statement

As the Reaper was the most prevalent and capable "hunter killer" aircraft [8], it is a logical starting point. Its mission was well understood and has predictable processes defined by the Find, Fix, Track, Target, Engage, Assess (F2T2EA) [18]. Additionally, its distributed digital control system and existing autonomous flight capability [23] means it could be automated without any physical modification. Surprisingly, there is almost no research into this sort of "ISR-centric" automatic mission execution that defines the hunter-killer mission. Thus, the problem statement is: Develop and implement a framework for the automated execution of the Find, Fix, and Track portions of the kill chain to serve as a starting point for future research.

1.3 Research Goals

The goals of this thesis are defined in terms of resultant behavior from the automated control system. They are designed so that the system's emergent behavior can be judged in both a simulated and real-world test flight comparatively. With this in

mind, the system should be able to:

- Fly an initial 360-degree orbit around a initial set of coordinates.
- While maintaining an orbit, build an internal three dimensional representation of the ground environment from the full motion video and telemetry taken in.
- Identify a particular object of interest and determine a hold area that allows good distance, look angle, and prevents visual occlusion.
- Execute appropriate aircraft holding to maintain that area.
- Above all, maintain visual custody of the target.

1.4 Development Methodology

In order to actualize these research goals, AFFTRAC was developed in two phases: simulation and flight test. These phases were discrete instantiations of AFFTRAC with different inputs and outputs. However, they shared a common architecture. The remainder of this section details these instantiations at a high level. Specifics are in Chapter 3.

This common architecture can be seen in *Figure 1* in green. Whether in simulation or in flight test, the rough process of AFFTRAC was as follows:

- Start with a full motion video source of a target environment.
- **Generate Partial Structural Point Cloud** - Using pairs of sequential images separated by sufficient parallax from aircraft motion during orbit, create a partial structural point cloud representative of the building environment.
- **Determine Target Location** - Using the same images as the previous step, localize the target in the image and place the target correctly in relation to the partial structural point cloud.

- **Conduct Occlusion Detection** - Trace rays from the identified target (a person) against the partial structural point cloud and determine angles of visual occlusion.

- **Generate Holding Area Density** - Create a holding area composed of known good azimuths and radial ranges. The relative density of the holding area should drive the aircraft to stay at the radial mean and not favor any particular non-occluded azimuth.

- **Generate Turn Command** - Given the current aircraft location and a holding area density, use a greedy algorithm (quadrant analysis) to produce turn commands that will keep the aircraft in the holding area density.

- Enact those turn commands in the current environment.

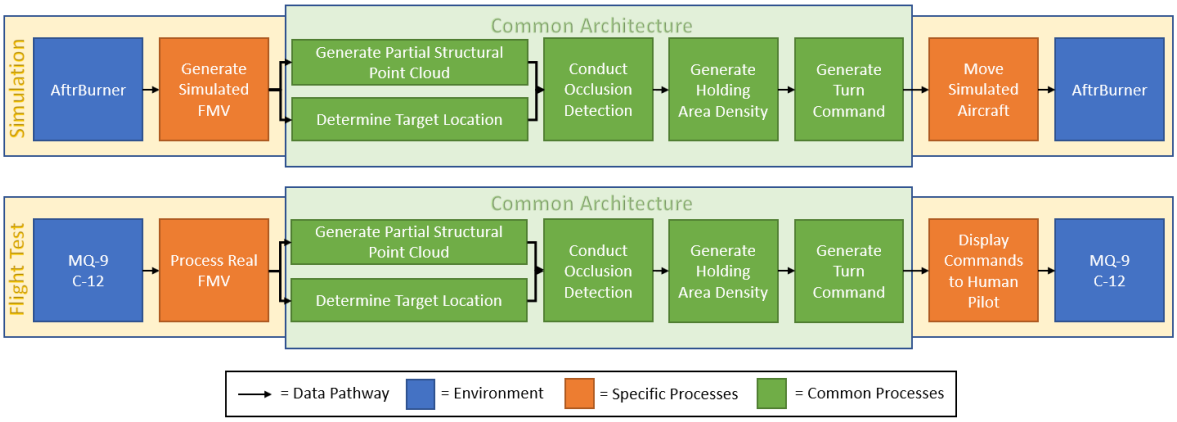

Figure 1: *AFFTRAC Common Architecture Overview*

AFFTRAC in simulation served two purposes. First, it allowed development in a controlled environment. Second, it served as a baseline for comparison to flight test. This was possible because all parameters could be controlled, all necessary data were available as a real-time input, and there was real-time control of the aircraft.

In flight test, multiple changes had to be made. These changes were necessary because real-time input (video and telemetry) and real-time control of the aircraft were not implemented. As a bonus, these changes additionally served as an exercise in risk reduction. This allowed component algorithms of AFFTRAC to be exposed to the realities of flight test and to determine where further development was needed. These components are further detailed in Section 3.6.

Two components were specifically tested: Generation of Holding Area Densities and Generation of Turn Commands. Additionally, the emergent behavior of the holds produced by following those turn commands over a period of time was evaluated as well. The flight test specific changes necessary to test these areas are outlined below:

- **Generation of Holding Area Densities** - Real-world full motion video (FMV) was collected from a sortie and then the telemetry and video was post-processed to produce a partial structural point cloud and produce a holding area density. This was done in post-processing rather than live as in simulation because AFFTRAC did not receive live video in flight. The intent for this was to evaluate how the resultant holding area density made during flight test compared to those produced in simulation.

- **Generation of Turn Commands** - Where the aircraft was in relation to the holding area density generated a turn command to the human pilot. The intent was to evaluate if the presented turn command always brought the pilot back into a logical holding position. This was only done in flight test.

- **Holding Performance of the Aircraft** - Because the partial structural point clouds could not be generated real-time during the actual sortie by the AFFTRAC image pair analysis algorithm, a commercial off the shelf (COTS) software Structure from Motion (SfM) algorithm is used in lieu. These high-fidelity and

complete structural point clouds were used during flight test in lieu of partial structural point clouds. The intent here was to evaluate how well the aircraft actually held in the best holding area density that AFFTRAC might produce.

1.5 Assumptions and Limitations

There are number of assumptions made in this thesis. The aircraft is assumed to be fully functional and have near-perfect positional awareness from GPS and other sensors. The aircraft is aware of its own position (altitude, ground position, airspeed, etc) and can position its Electro-Optical (EO) sensor to a location on the ground accurately. Thus, when the aircraft navigates towards a target location, there is no ambiguity from a positional perspective that it is in fact orbiting the correct target and is also "looking" at the correct location. These problems have already been solved as long as there is no GPS denial or system malfunction. GPS denied localization is another area of research well outside the scope of this thesis.

Additionally, this thesis assumes that target identification has been solved. For example, it is assumed that when targeting a single individual and given a full motion video stream that individual can be identified against a myriad of backgrounds and discerned from other individuals. Or in other words; it is assumed that the aircraft will have reliable knowledge about what specifically it needs to follow in a two-dimensional image. Thus, the aircraft will have an accurate location of what object it needs to keep in visual custody. The aircraft is also assumed to operate in a well-planned permissive environment. All of the necessary planning documents have been created and are accurate. Thus, when the aircraft is en-route to the target and arrives at the target location, it has complete freedom of movement relative to the target.

Finally, the flight test portion of this thesis has constraints and limitations different from simulation that are spelled out in the latter portion of Chapter 3. They are due

to the fact that real-time video and telemetry could not be acquired from the test aircraft nor could real-time commands be actuated through software to the aircraft. Additionally, there was noise in the location and sensed rotation of the EO sensor that were not specifically accounted for in AFFTRAC. This choice was made due to time constraints.

1.6 Thesis Overview

The structure of this thesis follows a typical order. However, it is important for the reader to keep in mind that there are two modalities: simulation and flight test. These two components will be addressed in parallel throughout the normal thesis progression. Chapter 2 consists of a literature and theory review. This chapter requires special attention from an academic reader as many of the operational concepts put forth are critical to understand. Several design choices flow from these operational paradigms. Chapter 3 details the methodology of both simulation and real-world instantiations. It is subdivided by components within those two domains. Chapter 4 discusses the results and evaluates their performance. Chapter 5 contains a qualitative comparison between simulation and test, overall conclusions, and specific recommendations for follow-on work.

II. Background and Literature Review

2.1 Overview

This chapter reviews necessary background subjects and related research in order to give the reader a foundational understanding. Section 2.2 introduces core concepts of offensive military air operations and planning. These concepts are necessary to understand how a UAV operates in the larger picture and why certain design choices are logical. Section 2.3 details the history of UAVs and how they evolved into their current role. Additionally, the MQ-9 Reaper is introduced and its capabilities are discussed. Section 2.4 covers how a specific mission set (ISR) is executed by an aircraft at a high level. Section 2.5 reviews image recognition techniques with an emphasis on reconstructing physical geometry. Section 2.6 provides necessary concepts for decision heuristics, a simple agent, and the creation of value maps.

2.2 Military Targeting and Prosecution

This section covers military air operations. It begins with an explanation of why Joint Doctrine is important and what is directly relevant within. It then transitions to the tactical level and the particulars of mission execution are discussed. Ultimately, this section explains the process of why an air asset is assigned to, and how it executes, a particular mission.

2.2.1 Joint Publications and the Joint Targeting Cycle.

For the operator, it is a given that a mission needs to be flown or that a task needs to be accomplished. However, when looking to automate part of this execution, it is necessary to understand the larger picture above the tactical level. For example, there are existing processes that manage target priorities, assign aircraft to targets,

and coordinate joint multi-asset efforts. These processes clearly define roles, how campaigns should progress, what is allowed, and provide standardized formats for information flow between processes. The Joint Publications (JP) provide this information. A detailed exploration of these publications can be referenced in *Appendix A*.

For the purposes of this thesis, a high-level understanding of two concepts from the Joint Publications is sufficient. The Joint Targeting Cycle is a process to address the concerns of the previous paragraph [18]. It translates operational-level objectives into executable missions with all the necessary support to do so. It develops targets, assigns assets, monitors execution, and integrates mission results with predictable progressions and standardized inputs and outputs from each step. The second is *battle tracking*. "Battle tracking is the process of building and maintaining an overall picture of the operational environment that is accurate, timely, and relevant [15]."

2.2.2 Joint Targeting Cycle and the Kill Chain.

The execution step of the Joint Targeting Cycle represents the actual sortie and tactical execution. Basically, it seeks to systematize this rough sequence of events: find a target of interest through various information channels, roughly geo-locate that target, get assets in vicinity to verify exact location of the target, visually acquire the target, maintain custody of the target while Positive Identification (PID) [15] is made and legal constraints are considered, kinetically engage the target, and observe after-effects [18]. This rough flow is formalized into discrete operational states via the Find, Fix, Track, Target, Engage, Asses (F2T2EA) or "kill chain" [15].

Figure 2 details the F2T2EA process:

Find is a generalized step of getting the aircraft and sensor payloads roughly in the area so they can collect data.

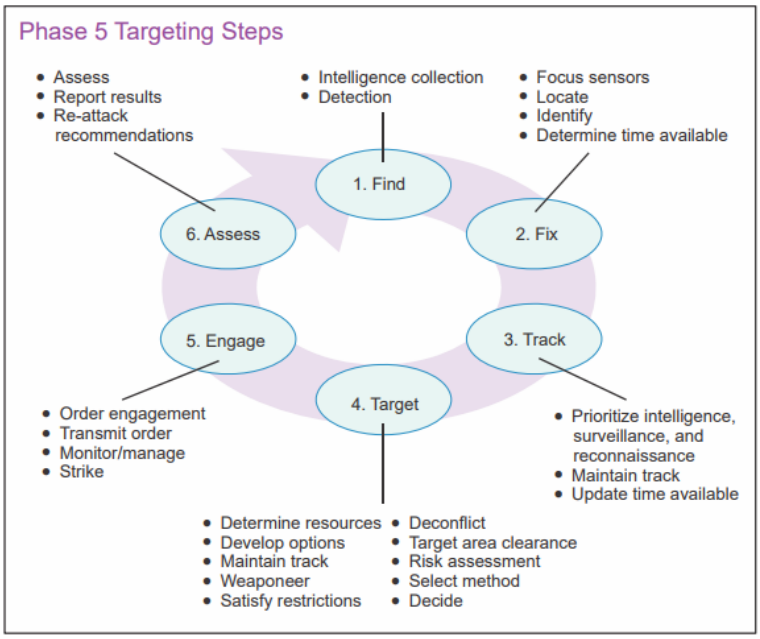

Figure 2: *The Find, Fix, Track, Target, Engage, Assess (F2T2EA) "kill chain" [18]*

Fix is localizing the target and establishing PID.

Track is keeping custody of the target and generating Patterns Of Life (POL) and other exploitable ISR and Signals Intelligence (SIGINT) data.

Target is when the target is evaluated for prosecution and applicable weapons are selected, proper aircraft position is established, and appropriate communications happen with controlling agencies.

Engage is when the aircraft physically executes the plan established in Target.

Assess passes kinematic effects to controlling agencies and determines whether to attempt a re-attack or monitor post-strike battle damage.

Additionally, it is helpful and appropriate to think of the transition between states as being triggered by certain actions. These "triggers" are defined later in Section 2.4 [18].

2.2.3 Battle Tracking.

As defined earlier, battle tracking is the process of creating an internal and functionally accurate representation of the events pertinent to execution of the tactical mission. Examples of things that a pilot would want to battle track include [15]:

- Location of friendly positions.
- Airspace and position of other air assets.
- Locations of enemy positions.
- Time-stamped location of previous takings.
- Brief descriptions of all above.
- Fire coordination measures (ground and air).
- Spatial layout of target area.

While this list is certainly not exhaustive, it creates a rough idea of what is meant by this internal representation. For the purposes of this thesis, the internal representation is limited to factors immediately relevant to tactical ISR. The items that are of concern specifically are: target location and description, spatial layout of target area, and holding area to conduct ISR.

2.3 Unmanned Aerial Vehicles

This section is intended to give the reader a conceptual understanding of Unmanned Aerial Vehichles (UAVs). The first subsection gives a brief historical context to give the reader a glimpse into how UAV's have historically been utilized. Next, the merits and capabilities of the MQ-9 "Reaper" are explored in order to convince

the reader that automating it is of immense value. Finally, an aircraft with a similar flight envelope (the C-12J) is detailed as it is used in the later flight test.

2.3.1 History of Unmanned Aerial Vehicles.

UAVs have a history in military operations that predates the inception of the US Air Force:

> "The origins of UAV development can be traced back to 1916 when Elmer Sperry of the Sperry Gyroscope Company (along with Peter Hewitt) successfully developed an automatic control system for the Curtiss Flying boat ... During WWII, the Germans developed a simple unmanned aircraft known as the V-1 Buzzbomb. Widely heralded as the first successful cruise missile, thousands of these unmanned aircraft performed one-way missions ... The modern UAV era began in late 1959. Prompted by the shoot-down of Gary Francis Powers U-2 over Russia, the first modern UAV efforts consisted of placing cameras in target drones ... The first substantial use of UAVs in the United States was during the Vietnam War. From 1964 to 1972, over 5,000 Americans lost their lives in SE Asia in downed aircraft and nearly 90% of the American POWs were pilots or crewmen who had been captured. Unbeknownst to most, during the same time period, over 3,400 combat UAV sorties were flown by SACs 100th Strategic Reconnaissance Wing over North Vietnam, China, Laos, and other locations in Southeast Asia ... The Israeli-made Pioneer UAV flew over 300 combat reconnaissance missions with the U.S. military during Persian Gulf operations in 1990-91. The RQ-2A Pioneer system received extensive acclaim for outstanding performance by Army, Navy, and Marine Corps commanders for its effectiveness as a Reconnaissance/Surveillance/Target Acquisition (RSTA), Naval Gunfire Support, BDA, and battlefield management platform ... The RQ-1 Predator UAV is manufactured by General Atomics and first flew in 1994 ... the Predator can conduct multifaceted roles over the battlefield ... When its not firing Hellfire anti-tank missiles at the enemy, the Predator uses its powerful surveillance cameras to give the theater air component commander continuous real-time surveillance of the battlefield. [11]"

2.3.2 Why the MQ-9 "Reaper".

The replacement of the MQ-1 "Predator" is the General Atomics MQ-9 "Reaper" remotely piloted aircraft . The Reaper is more capable in almost every way and silently became the workhorse of the Air Force. The main strengths of the Reaper are its cost efficiency [8][57][34][62], ability to amplify effectiveness of human effort, and digital extensibility [23]. As of November 2017, the USAF's Reaper fleet of 218 [37] aircraft has flown over 2 million hours in the previous 8 years [8]. Compare this to the F-16 at the time of writing, which was the largest single inventory of any plane in the USAF at almost 1,000 aircraft [37]. The F-16 flew just 1.6 million hours in the previous 8 years [57]. In 2018-2019, the MQ-9 was averaging 37,000 hours a month [8] versus the F-16's 17,000 [57]. The MQ-9 represented only 2% of the aircraft fleet [37] but accounted for 10% of the total flying hours. The F-16 represented almost 10% of the fleet [37] but only 5% of the flying hours. The Reaper flew more hours, both in total and per capita, than any other aircraft in the USAF inventory. And, if acquisition continues at current rates [39], it is probable that RPA flight hours will be the majority of flight hours by 2030.

Not only did the Reaper produce a large number of hours per aircraft, but it was also significantly cheaper to operate and more maintainable. The RPA training

Figure 3: *MQ-9 Reaper [23]*

pipeline is accomplished in under a year and for around $785,000 [34]. Compare this again to an F-16. For a F-16 pilot to reach the same level of mission readiness it requires $11,000,000 and over two years of training. There are now more pilots trained as Reaper pilots than for any other aircraft [51]. And even with this fact, the average Reaper pilot flies over 50 hours a month while the Air Force average is 17 [50]. Additionally, the cost per flight hour ($4,500) is an order of magnitude less than their manned counter parts ($20,000 - $160,000) [62]. Furthermore, with the retirement of the MQ-1B "Predator" (another General Atomics RPA) in 2018, the Reaper has the highest up-time of any aircraft at 90% versus 65% for an F-16 [37].

Perhaps the most interesting thing about the Reaper is not what it currently is, but what it has the potential to become. The Reaper's capabilities were already vast, but it is more extensible than any other aircraft. Because the missions are executed over KU-Satcom [23], what can become part of the weapon system is virtually unlimited. Unlike manned systems where the control and integration is ultimately internal and piecemeal, the only limit to how well integrated a new component can be is satellite bandwidth and software. Gone are weight requirements, fuel tradeoffs, creating custom hardware to interface with existing hardware, etc. The effectiveness of the plane can now be amplified by off-board processing, distributed Command and Control (C2), and Multi-Domain warfare [54] real-time.

Current efforts at General Atomics recognize this fact. Next generation block aircraft and ground control stations (GCS) seek to automate and integrate more effectively. The GCS seen in *Figure 4* is the legacy version. The Advanced Cockpit GCS "offers significantly improved situational awareness and reduced pilot workload. Innovations include intuitive interfaces that are designed to make potentially hazardous situations easier to identify and to improve the decision-making process generally. [6]." This is done in a variety of ways: Multi-source data is fused into a single common

operating picture on a single display. Integrated and automatic collision and terrain avoidance. Quick and intuitive switching between manual and automatic control. [6].

2.3.3 Reaper Capabilities.

The Reaper has a 66 foot wingspan and is 36 feet long, making it slightly larger than an A-10. It is capable of holding 3,000 pounds of external payloads on 7 hard points. The standard combat load is a combination of four "Hellfire" laser guided missiles and two GBU 500-pound bombs. The aircraft has a fuel capacity of 3,900 pounds giving it a burn rate of about 150 pounds per hour at the 27 hours max endurance [23]. The aircraft can either be remotely piloted from a Ground Control Station (*Figure 4*) or can fly fully autonomously. Remote pilotage can be done over C-Band Line Of Sight (LOS) or KU-SATCOM data link control. The Reaper has a belly-mounted Eletro-Optical InfraRed (EO/IR) Raytheon MTS-B/ANDAS-1, an internal Lynx Multi-Mode Radar (SAR, GMTI, and MWAS capable [7]) for all weather imagery, an integrated SIGINT/ESM system, and a communications relay.

The primary sensor of the MQ-9 "Reaper" is the ANDAS-1. The ANDAS-1 is a belly-mounted turret assembly manufactured by Raytheon and is a

> "Multi-Spectral Targeting System with Next-Generation Accuracy ... The new MTS variant allows mission commanders to use high definition data from an airborne tactical sensor to identify and engage targets with much greater accuracy, significantly improving overall mission effectiveness. The DAS-4 incorporates other major improvements, including: four high definition cameras covering five spectral bands; a three-color diode pump laser designator/rangefinder; laser spot search and track capability; automated sensor and laser bore sight alignment; three mode target tracker; and built in provisions for future growth [53]."

In other words, it is a very capable system for visual acquisition of ground targets from a considerable distance. "The field of regard for MTS-B effectively extends 360°

in azimuth and down to 90° in elevation (straight down), but in practice, such viewing angles are discouraged because they increase likelihood of losing the target [41]."

The Reaper is unique in its ability to persist and execute the entire air mission. It currently has a maximum loiter time of 27 hours [23] with a newer revision pushing that to 40 hours [22]. And, with a distributed control system that allows multiple crews to fly the same mission, human limitations on performance are removed. When this Time On Station (TOS) is coupled with the inherent abilities of the aircraft (Intelligence Surveillance and Reconnaissance(ISR), real-time Processing, Exploitation, and Dissemination (PED), and striking ability [23]) the end result is a "hunter-killer" asset capable of executing the entire "kill chain" by itself. These capabilities make the asset a "must-have" for basically any current operation regardless of mission type [41].

2.3.4 C-12J as a Reaper Analogue.

The C-12 "Huron" was manufactured by Beechcraft for the US Air Force. It was designed for passenger and light cargo transport. The C-12J variant has a significantly longer fuselage, slightly longer wings, and larger engines. The C-12J used in this test (callsign "Mabel") was a singular asset further modified. It has been altered with hot-swappable mounts for test racks, an additional generator for auxiliary power supply, two under-fuselage mounting racks for payloads up to 1,100 pounds in combination, and associated data cabling throughout the aircraft.

The flight envelope of the C-12J is similar to the MQ-9 in terms of max altitude, flyable air-speeds, and angle of bank in the turn. This makes it a perfect MQ-9 analogue. Additionally, the fact that it is a manned aircraft allows certain advantages for software development: Equipment can be carried onto the aircraft. For instance, this allows an independent location source to be easily acquired and fed into software without interfacing into the actual aircraft. Additionally, using a separate aircraft

Figure 4: *Ground Control Station (GCS) [6]*

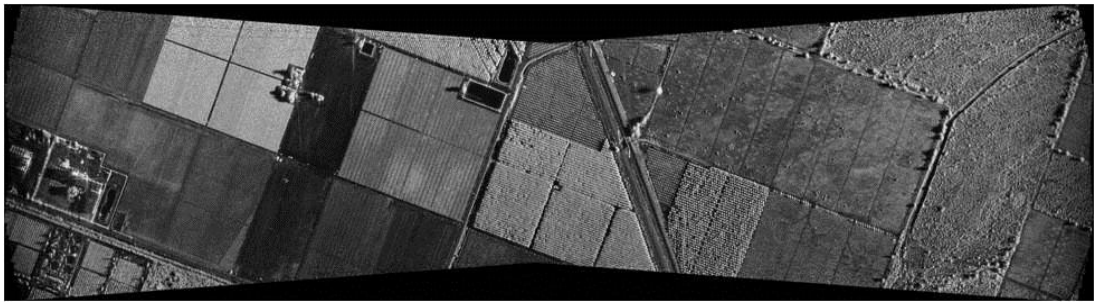

Figure 5: *Lynx Synthetic Aperture Radar (SAR) 1 Meter Imagery [7]*

Figure 6: *C-12J "Mabel" Aircraft*

with different handling characteristics and using a different imaging sensor (Lockheed Martin Sniper Advanced Targeting Pod (ATP)) serves to provide some evaluation of robustness of any algorithm that seeks to conduct automated ISR.

2.4 Conducting ISR

This section goes into more detail regarding the first half of the kill chain. The first subsection expands upon the concepts of Find, Fix, and Track with specific sub-tasks, considerations, and triggers to move between them. Next, basic aircraft holding that is useful for any aircraft is laid out. Finally, ISR-specific considerations and execution for the first two subsections are detailed.

2.4.1 ISR Expanded.

The first three steps (Find, Fix, and Track or F2T) can also be loosely equated with Intelligence, Surveillance, and Reconnaissance (ISR) [17]. Historically, there was not a single asset that was routinely capable of executing the entire F2T2EA chain [41]. However, with the advent of hunter-killer aircraft, all of this can be done continuously in a single node [41]. For the purposes of this thesis, transitioning between the phases represents an increase in two related confidence intervals. First, the confidence that a target has been found and well localized. Second, that the environment is understood well enough to allow time for localization and tracking.

The Find phase has a fuzzy beginning as it can start in a number of ways. The Joint Targeting Center can provide general location and objective through an Air Tasking Order [18]. Targets of opportunity can arise during ISR operations on a different target and change the aircraft's tasking real time. A previous execution cycle may necessitate a new rapid F2T2EA due to kinetic after-effects. Regardless of what initiates the kill chain, the point of the find step is fairly conceptually simple.

"During this step, emerging targets are detected and characterized for further prosecution [18]." *Figure 7* shows internal (grey) and external (white) considerations to this process and *Figure 8* shows an action mapping based on certain triggers. Simply put, once the target is classified as a potential target the aircrew transitions to the Fix phase.

Fix is much more well defined than the Find phase by its very definition. "The fix step begins after potential targets requiring dynamic targeting or on-call targets for deliberate targeting are detected. When a potential target is identified, sensors are focused to confirm target identification and its precise location. The correlation and fusing of data confirms, identifies, and locates the target [18]." Upon initial reading it may seem spurious that a target could be found at all without focusing the sensors on the target location. However, the gray area lies in the generalization of the word "sensors" and what "focusing" them means. This focusing is intended to facilitate the concept of PID. "PID is an identification derived from observation and analysis of target characteristics including visual recognition, electronic support systems, non-cooperative target recognition techniques, identification friend or foe systems, or other physics-based identification techniques [18]."

For the purposes of this thesis where the EO/IR camera is the only input, the sensor would have been focused (in the sense of a camera lens) in the Find phase. However, fixing the target requires various other considerations. Assuming the target is an individual, first determine where one could be logically. If the potential target is a building and no one is outside the building on arrival, it is logical to find the doors of the house and watch those. This is the highest probability of intercept and the task is now waiting for anyone to enter or exit and characterize them. Additionally, if seeking to watch the individual long enough to characterize them for PID, the aircrew must have a working knowledge of the environment so that the target does

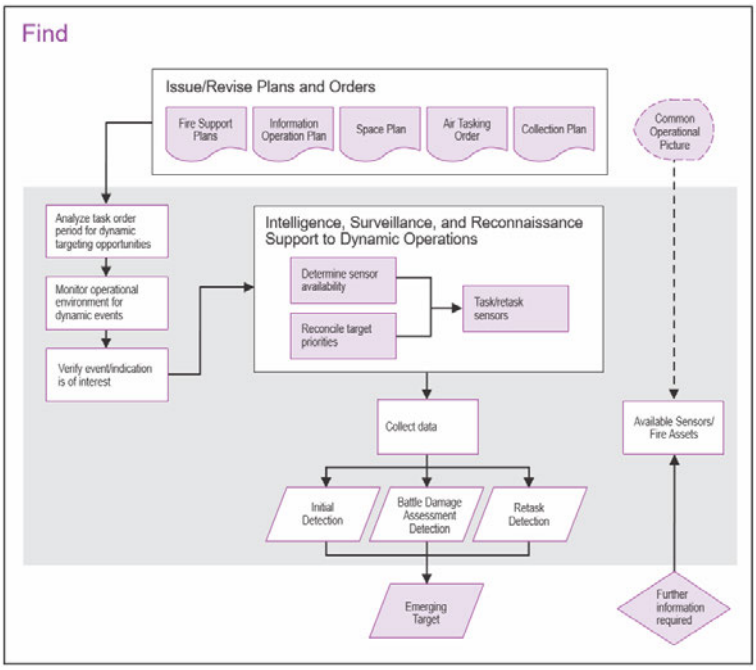

Figure 7: *Step 1 - Find [18]*

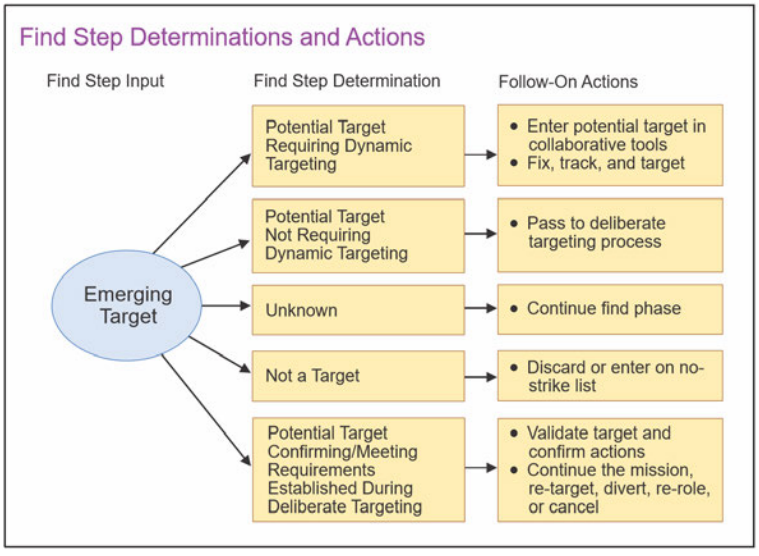

Figure 8: *Step 1 - Find (Actions) [18]*

not become visually occluded for extended periods of time. Thus, for an aircraft using a EO/IR visual sensor, the Fix stage becomes primarily an exercise in angles and aircraft placement. PID is the trigger to change the Track step.

If environment analysis is done correctly in the Fix step, the Track phase should be a seamless continuation of positioning the aircraft to maintain visual custody. "The track step begins once a definite fix is obtained on the target and ends when the engagements desired effect upon the target is determined. Note that some targets may require continuous tracking upon initial detection as an emerging target. [18]" The main objective during the Track phase is to not lose the target. "If track continuity is lost, the fix step will likely have to be repeated. [18]" Additionally during the Track step, potential areas of engagement or "target window of vulnerability [18]" are identified. Tracking these potential engagement zones and evaluating their worthiness create areas for the last three steps of the kill chain to occur. Again, for the purposes of this thesis this problem is primarily a question of aircraft placement and projecting good future hold locations. This paper does not seek to identify engagement zones or make the transition.

One last important ISR concept is Military Grid Reference System (MGRS). MGRS functions similarly to Lat/Long coordinates but with a couple important differences. First, it is designed to be more readable/transmittable to human and specifically machine end-users. "To facilitate machine-to-machine communication, an MGRS string is to have no intermediate spaces or punctuation marks and all the letters are to be capitals [64]." Secondly, it is metric and readily convertible into understandable and standard units. "For convenience, the world is generally divided into 8° of latitude by 6° of longitude, each of which is given a unique identification [DDA], called the Grid Zone Designation. These areas are covered by a pattern of 100,000-meter squares. Each square is identified by two letters [AA] called the

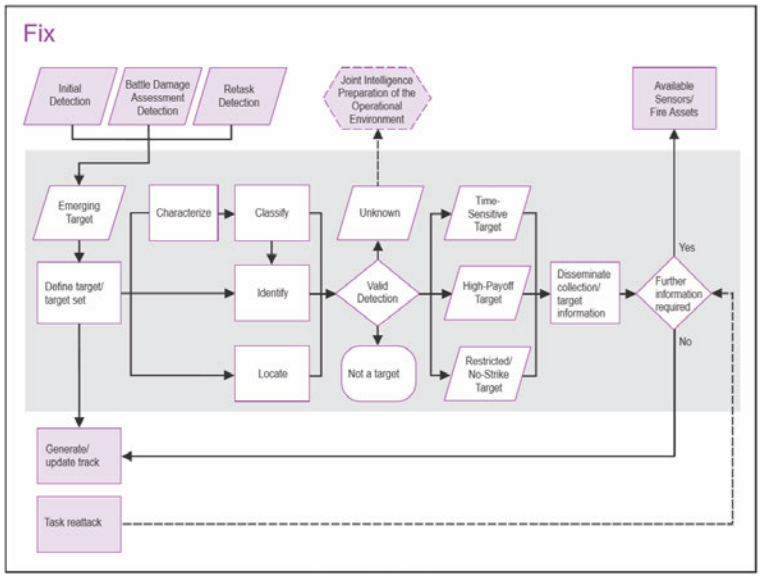

Figure 9: *Step 2 - Fix [18]*

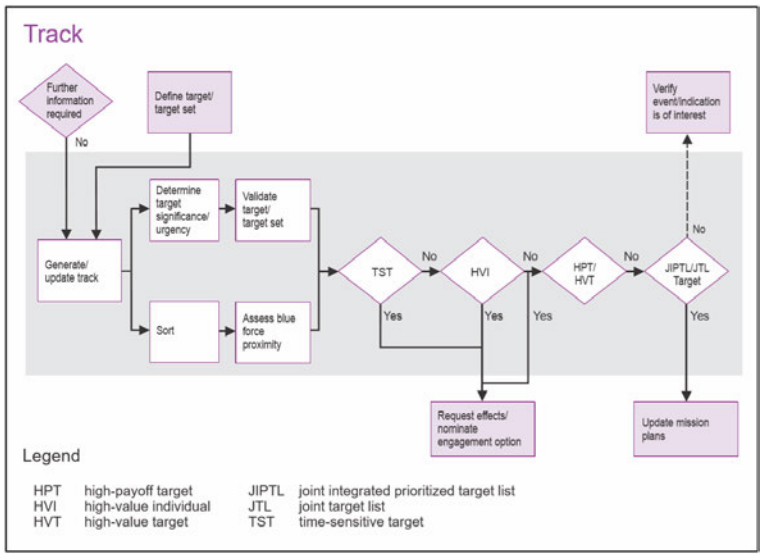

Figure 10: *Step 3 - Track [18]*

100,000-meter square identification. This identification is unique within the area covered by the Grid Zone Designation [42]." Thus, the first 5 characters (Grid Zone Designation and square identification) in the string are in the format [DDA AA]. The next 10 digits represent "easting" and "northing" from the lower left corner of that Grid Zone Designation.

2.4.2 Aircraft Holding.

As mentioned in the previous sub-section, traversing the steps of F2T with an airplane seeking to visually exploit an emergent target is primarily a question of holding. Three types of aircraft holding are detailed: The wheel hold, the sector hold, and the tight or "jelly-bean" hold. For simplicity's sake they are presented in a general fashion and wind's effect on the hold is qualitatively mentioned. The wheel hold is a simple hold and can be conceptualized as arcing around a target. Given no wind and a static target, a given radial distance can be maintained with a fixed Angle Of Bank (AOB). In practice this is never achieved. Variations in control systems, winds, human attention, etc produce a need for constant AOB refinement. A circle hold can be seen in *Figure 11.*

Sector holding allows the aircraft to remain roughly at a given radius within a restricted heading relative to the target. It is desirable over a "race-track" because with a wide enough sector, a fair amount of time can be spent arcing around the target at a constant distance. The effect of wind on AOB can be seen in *Figure 12.* Tight holding is the same concept as sector holding except that the restricted relative heading is tight enough that there is no space to establish an arc and reverse the holding turn. Instead, the aircraft essentially stays in a constant turn attempting to maintain a fairly constant hold radius and stay within the heading constraints. It is also called a "jelly bean" hold because wind causes the plane to drift during the 360.

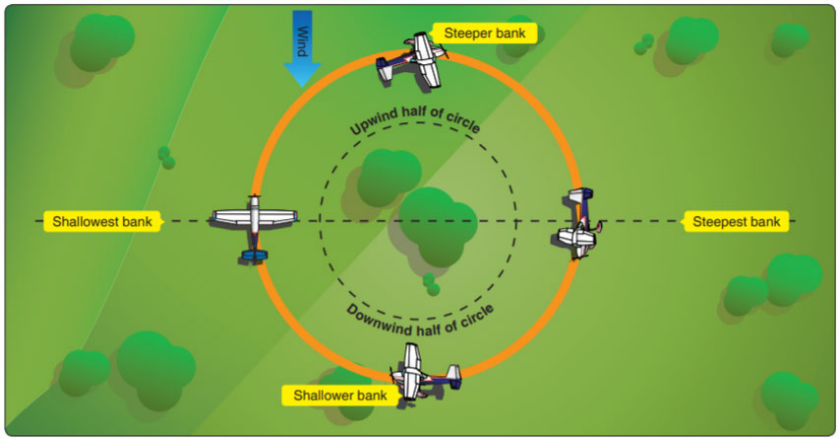

Figure 11: *Circle Hold [1]*

This can be seen in *Figure 13*. In order to counteract this, when the aircraft is facing the wind, the turn is let out to re-establish the desired hold location.

2.4.3 ISR Specific Holding.

ISR holding combines the processes and aircraft holds detailed previously. ISR specific considerations arises from the actual sensor utilized and physical flight characteristics. The flight characteristics of an aircraft need to be considered only in the way they drive hold distances. Fortunately, these hold distances can be derived fairly easily. In order to get the best possible picture for any image system, decreasing distance will increase fidelity. However, this has to be balanced with the ability of the aircraft to maneuver. Therefore getting the aircraft as close as possible to the target while maintaining a margin for maneuvering is the main objective. As stated earlier, an EO/IR sensor has an increased possibility of losing the target when the relative nadir of the aircraft approaches 90 degrees. Thus, the aircraft needs to remain far enough away so that banking away from the target will not result in the relative nadir getting near the target.

Assuming a Standard Rate Turn (SRT) of approximately 25° AOB and at 20,000

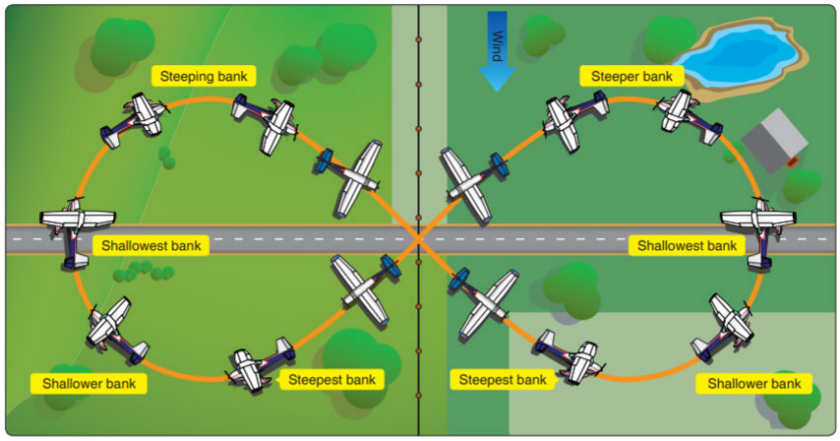

Figure 12: *Sector Hold [1]*

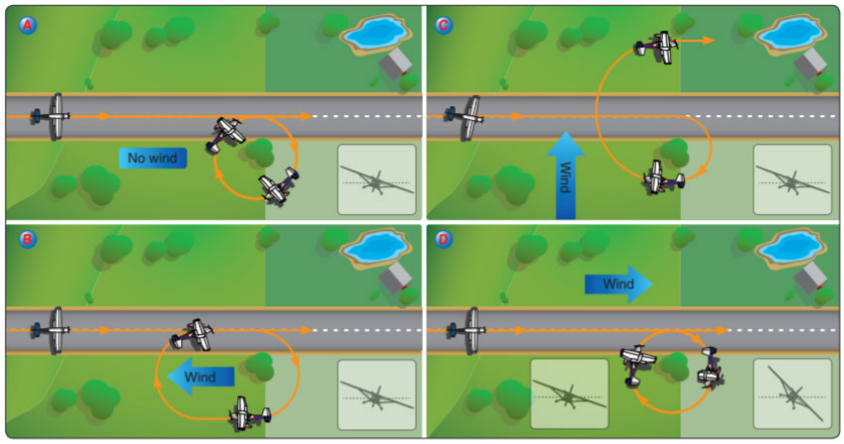

Figure 13: *Tight or "Jelly Bean" Hold [1]*

ft height above target (HAT), this results in a ground distance of 2.58 kilometers. A SRT is defined as a turn that changes the heading of the aircraft at a rate of 3 degrees per second.

$$sin(25^\circ) * 20,000ft * \frac{0.0003048\ km}{1\ ft} = 2.576\ km \tag{1}$$

This estimate is padded by approximately another 2 kilometers to arrive at a minimum of 4.5 km. This allows more turn authority than a SRT ($\frac{3^\circ}{s}$) and gives half a kilometer margin of error.

$$sin(40^\circ) * 15,000ft * \frac{0.0003048\ km}{1\ ft} = 3.918\ km \tag{2}$$

The maximum hold distance then becomes a function of true airspeed combined with a SRT. At 140 Knots True Air Speed (KTAS) this is approximately a 3 km diameter.

$$\frac{140\ NM}{1\ h} * \frac{1.852\ km}{1\ NM} * \frac{1\ h}{3,600\ s} * \frac{1\ s}{3^\circ} * \frac{360^\circ}{\pi} = 2.7524\ km \tag{3}$$

Thus, the radial hold distance ranges from 4.5 to 7.5 kilometers. However, if able, spending time at a constant distance closer to the target is more optimal than just remaining within the derived range. The mean of this range was used as an optimal distance.

2.5 Extracting Geometry from Images

If ISR is to be conducted autonomously, the underlying three dimensional structure of the environment needs to be recovered. This section starts with basics of image processing and then moves to feature detection. Those features are then exploited in a stereo vision algorithm. Camera calibration is detailed as a way to increase the

accuracy of a stereo vision algorithm. This section finishes by outlining various ways to exploit the point clouds created from stereo vision.

2.5.1 Basics of Image Processing.

Any sort of visual data is ultimately be resolved into a series of digital images for storage. This holds true whether it is virtual or real in origin. The images are represented in a matrix (or two dimensional array) of intensity values. For most black and white images, each matrix value or pixel is represented as a 8-bit value between 0 (no intensity) and 255 (max intensity). For color images, each pixel is represented by separate channels at increased storage cost per pixel. An example of image channeling is Red Green Blue (RGB) decomposition. Here, again for most images, each pixel is a 24-bit value composed of three 8-bit channels. Intensity (or gray-value) can be derived in an RGB image by averaging the three channels [24].

While there are several basic image processing techniques (pixel-level transformations, histogram processing, arithmetic operators [24]) only two major areas are detailed specifically: Image processing in the spatial domain and the frequency domain. Both spatial processing and frequency filtering have their uses. Spatial processing is generally useful when phenomena need to be localized as the processing is done via convolution in the spatial domain. Examples include: edge detection, sharpening, blurring, thickening, dilation, erosion, and more [24]. Frequency is generally more useful when meta-characteristics of the picture need to be altered. Examples include high-pass filters, low-pass filters, homomorphic filtering, image sharpening, and edge detection.

Image processing in the spatial domain is primarily accomplished with convolution matrices, also called kernels or masks [3]. The general expression for an MxN image f with a mxn mask w where $a = \frac{m-1}{2}$ and $b = \frac{n-1}{2}$:

$$g(x,y) = w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x + s, y + t) \tag{4}$$

Above $g(x,y)$ represents the resultant value of each (x,y) pixel after convolution. An example of a mask is the Sobel filter [3] for horizontal edge detection (*Figure 14*). This mask is a *balanced mask* because the component numbers all sum to 0. After the convolution, the resultant image contains spatial information. For a Sobel filter, each pixels intensity represents how strong of a vertical edge transition has been detected in the 3x3 pixel space.

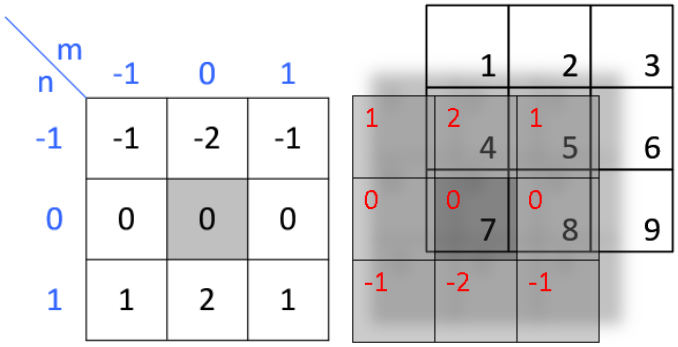

Figure 14: *Image Convolution [3]*

This can be normalized from 0 to 1 if a probability is desired or 0 to 255 to display maximum granularity to human eyes. The other type of mask is a *weighted mask* because the values do not sum to zero. A smoothing mask is an example [24]. The equation for that convolution can be seen below:

$$g(x,y) = \frac{\sum\sum w(s,t) f(x + s, y + t)}{\sum\sum w(s,t)} \tag{5}$$

Image processing in the frequency domain is useful in similar ways. The Discrete Fourier Transform (DFT) is defined by:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})} \tag{6}$$

The inverse is as follows:

$$f(x,y) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} F(u,v) e^{j2\pi(\frac{ux}{M} + \frac{vy}{N})} \tag{7}$$

When processing an image in the frequency domain the rough process is as follows: Multiply the image by $(-1)^{x+y}$ to center the transform to get $f(x,y)$. Take the DFT of the result to get $F(u,v)$. Multiply $F(u,v)$ by a filter function $H(u,v)$ to arrive at $G(u,v)$. Compute the inverse DFT of $G(u,v)$ to get $g(x,y)$. Multiply the real part of $g(x,y)$ by $(-1)^{x+y}$ to recenter the image [24]. An example of a filter function is a low-pass filter. In the shifted Fourier domain, low frequency information is central to the image and high frequency radially distal. By discarding all information outside a certain radius in the frequency domain, the retrieved spatial image is blurred because high frequency information is lost. This ideal is fundamental to lossy image compression [24].

2.5.2 Feature Detection.

Of particular interest from the previous sub-section is that characteristic information can be derived from either domain. The Sobel Filter [60] found and localized edges. Using a correctly constructed high-pass filter accomplishes something similar. Finding these artifacts is the fundamental idea of feature detection. Feature detection is broadly divided into three categories: Edge, Corner, and Blob. Each category is briefly explored in the following paragraphs.

Edge detection is usually accomplished by a series of convolution filters applied in sequence to identify edges at various angles. Examples include the Canny [27],

Sobel [60], and Prewitt [58]. The end result can be seen in *Figure 15*. In *Figure 15*, the angle of the edge detected determines the color of the edge line in the picture on the right. Edges can be used as features if they are properly filtered through further convolution techniques such as skeletoning or double confidence [58].

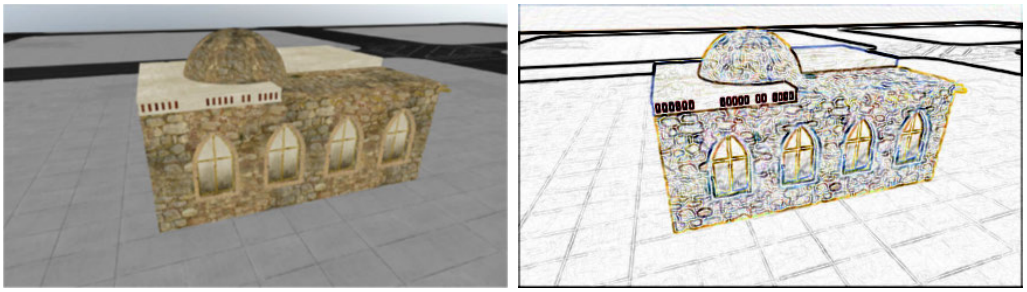

Figure 15: *Edge Detection - Sobel*

Corner Detection is done in a variety of ways. It can be divided into three main methods: Template based, contour based, and direct corner detection [49]. Older methods such as the Harris & Stephans [26] method use convolution masks as in edge detection and fall into the first category. Harris also used Canny edge detection and linked the found corners to create continuous edge segments. These segments were then auto-correlated to create rudimentary feature tracking [26]. A famous contour-based example is Frstner [49]. The area with the most current research is into direct corner detection with the classical example of the Smallest Univalue Segment Assimilating Nucleus (SUSAN) [49] corner detector. A more modern approach is Features from Accelerated Segment Test (FAST). They work in a similar fashion. At every pixel a circular mask is applied and the intensity values are categorized by relative brightness. If coherent sections (SUSAN) or a continuous circle (FAST) emerge, the pixel is classified as a corner. The end result of all the algorithms is roughly the same and can been seen in *Figure 16*.

The last general category is blob detection. Here, the regions or segments of

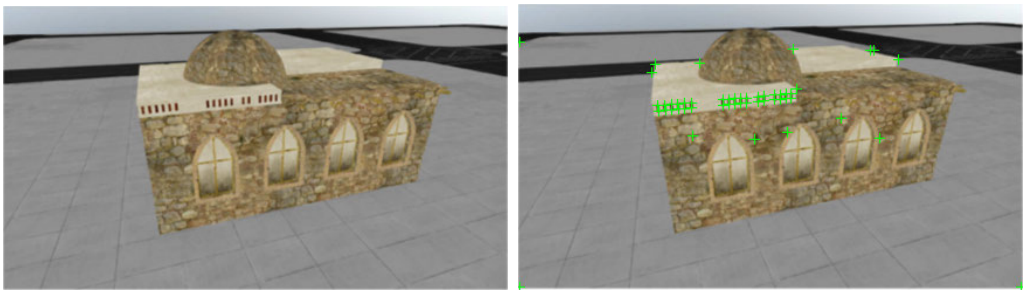

Figure 16: *Corner Detection - Harris*

the image are found. This can be done in a variety of ways. Previously mentioned convolution masks or frequency filters can be used to identify edges. After edge identification, image segments can be created in those regions defined by the discovered edges. Often there is a factor to determine how strong an edge needs to be to define a boundary, thus effecting how segmented the resultant image becomes [24] [32]. More modern approaches work in much the same way, but use more novel methods to arrive at more accurate edges. Principal Curvature-Based Region detector (PCBR) uses a difference between a gray-scale morphological close and an eigenvector flow hysterisis threshold to identify edges [16]. Maximally stable extremal regions (MSER) uses a pixel intensity threshold to classify individual pixels and the union areas with similar intensities [28]. In all cases, the blobs may be processed as is or they may be further processed to determine characteristics such as center of mass (erosion [24]) or attitude (skeletonization [24]) to localize the feature to a point or line.

Other methods of creating features can be more complicated or aggregative in scope. Larger gradient masks [24] or Difference of Gaussian can be used to identify slope and curvature as features [32]. Thresholding [32] and template matching [32] can also be used a higher level features. Additionally, features can be aggregated into meta-features [68] when appropriate. Features are used primarily as intermediaries in other exploitative algorithms. Examples for use include: camera pose

estimation [47][59], machine learning [68], image labeling [67], and image segmentation [24][32]. Relevant uses to this thesis are explored in subsequent subsections.

2.5.3 Stereo Vision.

Integrating two vantage points to recover three dimensional information is the principle idea of stereo vision. To do this a structure has pictures of it taken from two known vantage points. This is generally two distinct cameras with a known fixed distance between them. Feature matching is then done between the images to identify the same feature in these vantages. These are compared to a camera location or pose to reconstruct relative geometry. All of the features are thus localized in space and can be projected into a point cloud. Afterwards, image data may be mapped back onto the point cloud to create a textured 3D approximation of the original [14]. The primary advantage of stereo vision is that it can recreate three dimensional information for a specific moment in time whereas other methods require integration over a period of time.

The definition of a feature has been discussed previously. However, many of the previous feature detection methods are inadequate because they fail when the image scale and observed angle are radically different. Identified features thus need to be further characterized so they can be matched. The answer is a Scale Invariant Feature Transform (SIFT). The steps are as follows [38]:

Take 16x16 square window around detected interest point (8x8 shown below).

Compute edge orientation (angle of the gradient minus 90) for each pixel.

Throw out weak edges (threshold gradient magnitude)

Create histogram of surviving edge orientations (8 bins)

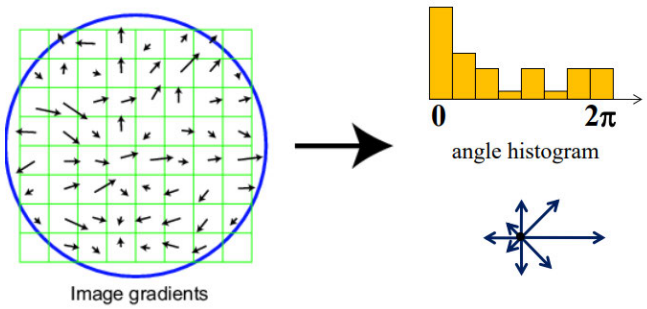

Figure 17: *SIFT Transform [38]*

Divide the 16x16 window into a 4x4 grid of cells (8x8 window and 2x2 grid shown below for simplicity)

Compute an orientation histogram for each cell

16 cells * 8 orientations = 128 dimensional descriptor

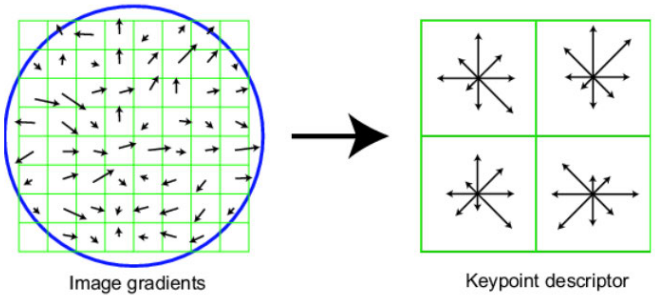

Figure 18: *SIFT Transform [38]*

Features then can be matched from image to image. SIFT is capable of extraordinarily robust matching with up 60° of vantage rotation, significant changes in illumination, and can run in real time [38]. After the candidate matches are confirmed, locality of the features is compared to further ensure matching of features. The next step is determining the pose of a camera. Feature matching can be used to determine pose if necessary but for simplicity it is assumed to be known here. Thus, with

known relative geometry between matched features in the 2D images and 6 degrees of freedom camera pose information, the geometry of the structure can be recovered and projected into 3D space as seen in *Figure 19* [70].

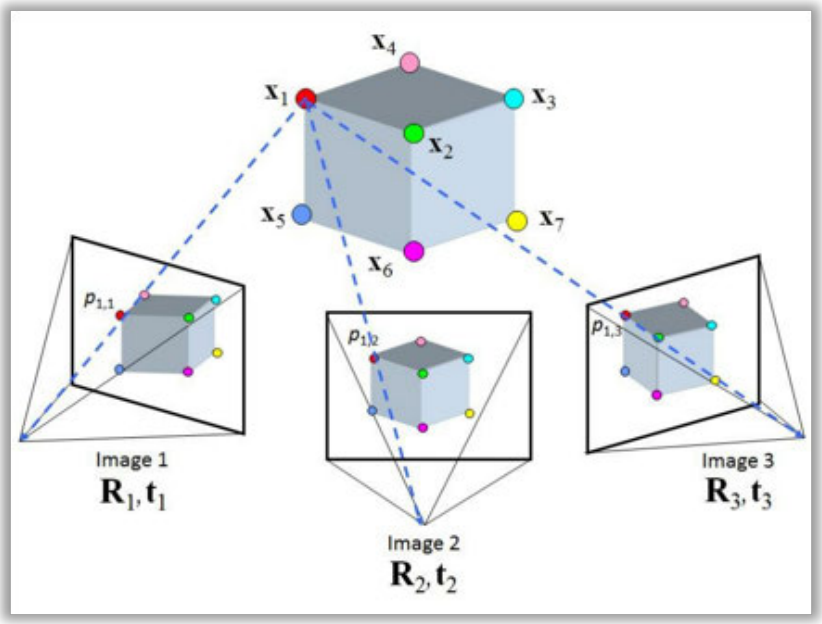

Figure 19: *Point Projection [38]*

The end result is a localized point cloud of features that approximates the actual structure captured in a pair of 2D images. However, generating the point cloud through stereo vision effectively is a continuing area of research in of itself [33]. Additionally, components during the project process may be used for different ends. For example, if the camera motion is known to be sequential along a single axis, the relative motion of the features can be use to classify planes by relative shared vector [43].

However, in relation to fixed-wing ISR, there is limited research into recovering 3D spatial information from two dimensional images. Stereo vision on large fixed-wing ISR cannot be done in usual manner. This is because the aircraft is not large enough to have two cameras with sufficient distance between them for a ground target. The

distances for 5 degrees of parallax is approximately 350 meters at 4.5km ground distance. There is research into monocular systems such as Simultaneous Location And Mapping (SLAM) [29][69] in small UAVs. This would allow on-line three dimensional reconstruction with an integration time acceptable to USR. However, there is no research into implementing a system like this into a large fixed-wing ISR asset.

Structure from Motion (SfM) in large fixed-wing ISR [35] is another way of recovering three dimensional from a series of images. While the reconstruction is of much higher quality than any other method described so far, it comes at the expense of integration time. Dozens of images and hours of processing are necessary and this makes SfM fundamentally unsuitable for tactical ISR. Thus, there is no existing method that can be used and one must be created. Chapter 3 details how the stereo vision is modified to become disjointed and monocular.

2.5.4 Camera Calibration prior to Three Dimensional Reconstruction.

The first step in exploiting a two dimensional image with the intent of recovering three dimensional spatial information is to calibrate the camera. This is accomplished by taking a series of pictures of an object with known geometric measurements in various orientations. The most commonly used object is a checkerboard as it has known consistent dimensions. Images are subjected to a convolutional filter to pick out the corners in the checkerboard.

Thus, it is possible to programmatically determine the relationship between coordinates in an image and the actual two dimensional spatial coordinates [71]. This is a necessary prerequisite of recovering three dimensional information as it is necessary to know that the 2d spatial information is correct. The end result of this process is what is known as a calibration matrix ("K-Matrix") or a "Distortion Matrix". The K-matrix transforms sensed image data to "spatially accurate" image data. Spatially

accurate data is what the sensed image would be if there were no imperfections or distortion from the imaging system. The inverse K matrix does the opposite. In the following equation f_x & f_y represent the camera focal length, s represents the camera skew, and x_0 & y_0 represent principal point offset.

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

Thus, the relationship between image coordinates (x_n & y_n) and spatial coordinates (x_p & y_p) can be modeled. The z-coordinate determination is in Section 3.3.3.

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = K \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} \tag{9}$$

The former equations model a perfect system. The curvature of the lens and irregularities in the imaging system introduce localized distortion that needs to be corrected as well. This is essentially a series of Zernike polynomials [44] where $k(1)$ through $k(5)$ are distortion parameters. This is the confusion matrix and converts x_n & y_n into x_d & y_d

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = (1+k_c(1)r^2+k_c(2)r^4+k_c(3)r^6) \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} 2k_c(3)x_n y_n + k_c(4)(r^2+2x_n^2) \\ k_c(3)(r^2+2y_n^2) + 2k_c(4)x_n y_n \end{bmatrix} \tag{10}$$

Thus, the x_p & y_p coordinates are instead derived by multiplying the K matrix with x_d & y_d:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = K \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{11}$$

2.5.5 Point Cloud Exploitation.

Regardless of the environments or assumptions that go into generating the point cloud or how it is exploited: it is clear is that using feature detection to generate a point cloud is a fairly robust strategy. Once a point cloud is created, the data can be processed in a variety of ways to extract geometry from it. One possibility is to load several basic 3D models of expected structures. These structures are compared to the generated point cloud to generate a Root Mean Squared Error (RMSE). In the following equation M is the number of points in the reference model and N is the number of points in the generated model:

$$RMSE = \sum_{m=0}^{M} \sum_{n=0}^{N} \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2 + (z_m - z_n)^2} \tag{12}$$

The magnitude of the value represents how close a match the two models are to each other. The computational complexity of this can be further reduced with assumptions about the environment. For example, if trying to derive geometry from a house, it may only require rotation about the z axis (up and down) as the house is assumed to be upright and level [13]. Additionally, the derived model of the point cloud can be down-sampled by throwing away random points but still maintaining structural integrity. Once the model is matched, the most simplistic "wire-frame" representation of the known model may be used to represent the scene.

Similarly, it may be useful to try to detect planes within a point cloud. Using a Random Sample Consensus (RANSAC) [21] approach, various random combinations

of points in the cloud are evaluated for how well they fit an average plane between the points. This can been seen in *Figure 20* [5]. Once a series of plane have been detected, the intersection of the planes can be assumed to represent logical edges and vertices as appropriate. The simplified geometry can thus be extracted with little knowledge of the ideal model. As before, assumptions about the nature of the scene being observed increase the chance of success [13] [5].

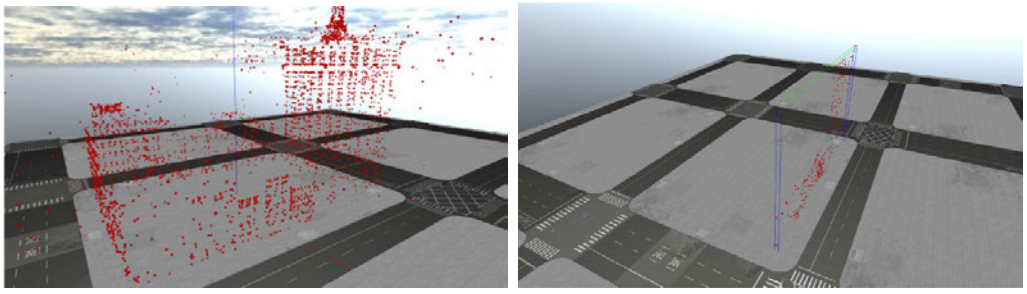

Figure 20: *City Block - RANSAC Plane Detection*

Another possibility is to use points clouds of known models use Minimum Volume Bounding Box (MVBB) [9] analysis to compare the known model to the derived model [21]. Similarly to plane detection, a RANSAC approach is used to compare the point cloud to known geometric shapes. Rather than a RMSE analysis a MVBB approach to find the shape with the smallest volume that contains all of the points. Again, assumptions about physical structures on the ground can further improve this process. Additionally, as this thesis is concerned with approximate geometry, having the bounding shapes overlap is not a constraint [21]. A final option is to use the point cloud directly to represent the structure. It may be prudent to consider outlier rejection through geometric bounding or random down-sampling. This may be useful if the previous methods are having limited success, are computationally too expensive, or the structure too irregular to fit into basic geometric concepts [21].

There are examples of small UAVs using structural point clouds in navigation [72].

The point clouds are primarily used to represent obstacles to avoid in path planning. However, there is little research into using them in a fixed-wing ISR scenario. In large fixed-wing ISR, they are more important as visual occlusions than something to be physically avoided. Most existing research into automating ISR has to do with path planning between disparate targets [10] or automatically generating intelligence from images. The latter is generally done with Convolutional Neural Nets (CNNs) [10] but requires humans control aircraft based on the information found. Exploiting the structural point clouds autonomously is a gap in research. AFFTRAC attempts to fill this gap by translating the point cloud into a value map as described in Section 2.6. The exact implementation is in Chapter 3.

2.6 Decision Heuristics

This section covers the high level concept of what to do with structural information once it is generated. It begins by detailing how a simple agent works and what information it needs to function. Next, the idea of a value map is discussed as it provides a more useful framework for an agent to exploit. Afterwards, a statically generated value map is discussed as a computationally cheap way to achieve the former. Finally, a method of generating something approximating a statically generated value map via ray tracing and occlusion detection is discussed.

2.6.1 Simple Agents and the Markov Assumption.

The previous section of this chapter covered how to reduce image data into an internal three dimensional representation. In order for an aircraft to act autonomously, it needs to determine what to do with that information. Some algorithm is needed to translate information into controlling action. The concept of an "agent" is one such algorithm. "An agent is anything that can be viewed as perceiving its environ-

ment through sensors and acting upon that environment through actuators. [56]" A flowchart showing this idea can be seen in *Figure 21.* In this simple agent, there is some reward and some state at a current time step fed into the agent. Based on that information, the agent makes an action which changes its relation to the environment. The agent then considers the state and reward at the next time step to make the next action.

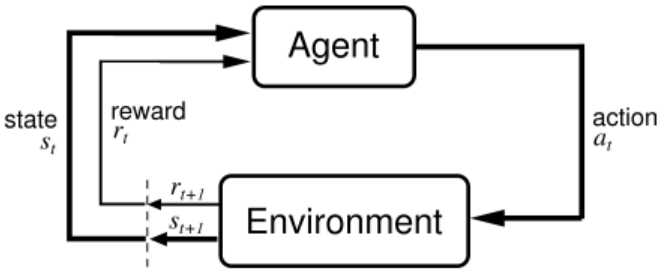

Figure 21: *Simple Agent Interactive with Environment [61]*

This agent is reactive in nature and doesn't plan out beyond the current time step. Additionally, it doesn't consider the past actions it has made to future actions. This may or may not be acceptable given the problem an agent is trying to solve. However, when it is valid, this is know as the Markov Assumption [61].

$$P(X_{t+1} = s_{t+1}|X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0) = P(X_{t+1} = s_{t+1}|X_t = s_t) \quad (13)$$

The Markov assumption states that the current state is sufficient to model the chain of past actions that led to this current state. The primary concern in this thesis is a high level control agent capable of executing across a simple set of instructions. For a constant velocity and constant altitude aircraft the instruction set is essentially: turn left, turn right, or go straight. If the agent has sufficient information in the current time step, it can perform this simple action. The Markov Assumption is

made for sake of computational and algorithmic simplicity. However, it often comes at the cost of the agent's ability to make better decision with more information over time. Regardless, in order to act the agent needs a way to rank possible action. The next subsection considers how to generate that information.

2.6.2 Reinforcement Learning.

Value maps (as seen in *Figure 22*) are generally used to determine action given a current state in mobile robotics [63] with localized rewards. For example, a robot may wish to determine where it should go and the shortest way to get there. However, determining that requires the robot to run numerous simulations going to all possible locations in all possible sequences to find an optimal answer. While there are multiple methods to reduce the computational requirements [63] of this task, ultimately the robot will end up with a place to be and a means to get get there. Value maps store this information in a data structure for future reference.

The primary building block in a value map is a reward function as such as the one in *Equation 35*. Path planning can be defined by an "expected cumulative payoff [63]" based on a series of a actions. As the agent navigates the value map through N time steps discounted a rate of μ, all possible s states are aggregated to produce the relative value of that course of action. The only term not addressed is $p(s_j \mid a, s_{j+1})$. This represents a probability of action. In other circumstances this might model uncertain outcome given control input. In this context it is considered to be likelihood of action and each action is given an equal probability.

$$V^{avg}(s_0) = \mu \ * \ MAX_a[R(x,y) + \sum_{j=1}^{N} V^{avg}(s_j)p(s_j \mid a, s_{j+1})] \tag{14}$$

Another thing that can be done is to analyze a value map to produce an action at every state rather than just a map of possible rewards. This concept is a policy

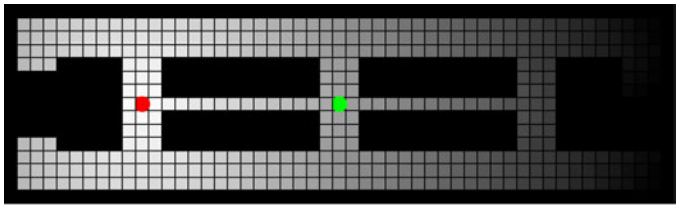

Figure 22: *Value Map [63]*

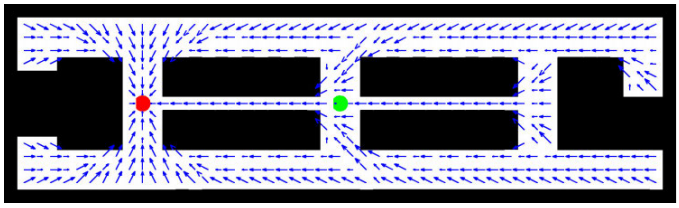

Figure 23: *Policy Map [63]*

map as seen in *Figure 23*. By iterating over each of the possible states continuously, a policy can be determined at every time step. The agent, when it arrives a current state, simply needs to consult the existing policy for that state. Here a policy function is considered. As the environment is assumed to be fully observable we can use the Markov Assumption [56] to generate a simple policy π.

$$\pi : s_t \rightarrow a_t \tag{15}$$

Here every state s at time t has an optimal action a. Thus, the value map can be overlaid with an optimal policy for every location.

$$\pi = ARGMAX_a[R(x,y) + \sum_{j=1}^{N} V^{avg}(s_j)p(s_j \mid a, s_{j+1})] \tag{16}$$

As a value map is first and foremost a reward distribution, all of the locations have an associated reward. Getting to an area of higher reward should correspond to a area of good location [63]. At any point, a policy can be derived even if that information

is not stored for the entire map. This is the fundamental idea behind reinforcement learning. It exploits a value or policy map to maximize reward over time [63]. The trade-off in using reinforcement learning is that actual optimal behavior is sacrificed for approximately optimal behavior. The return is speed of exploitation [63]. This is necessary for any non-trivial problem.

A fixed wing aircraft is unique in that it must constantly move even if it is in a good location and can only change its current heading so much. This means that the aircraft is optimizing the reward over a constantly changing hold rather than achieving a specific location. Small UAVs don't have this problem as they are generally rotary-wing and can hold a constant position. There is a large body of research in calculating an optimal path [36][31][48][12] for aircraft flight planning in small UAVs.

However, the algorithms used for small UAVs generally don't transfer to large fixed-wing ISR assets. The means of collection, speed, distances involved, and software radically differ. The algorithms developed for small UAVs can often not be used. Where reinforcement learning is used in large fixed-wing ISR, it is generally used for path planning and sensor coverage [10]. The agent is concerned with making a path to maximize intelligence quality over several targets in the shortest path or time.

There is no research into using reinforcement learning in a tactical ISR scenario as described earlier in this chapter outside of this author's previous work [46]. This work proved promising in a simplistic simulation and, while a simple agent, produced promising results. This thesis seeks to determine if that simple agent model is sufficient to produce good holding behavior in a more complicated simulation and actual flight test.

This chapter covered the military targeting and prosecution "kill chain", introduced the MQ-9 Reaper, and explained how basic ISR functions. This provided the background for how human operators currently execute the ISR mission and out-

lined the process to automate. The next two sections reviewed literature looking for ways to build internal representations of the environment necessary for automation. This identified the algorithms responsible for recovering three dimensional information from two dimensional images and exploiting value maps to optimize total reward. However, these algorithms are not well developed in the large fixed-wing ISR arena. Additionally, when they have been adapted, the are generally used for path planning rather than tactical execution. These concepts are modified and adapted to automate the F2T mission in the next chapter.

III. Methodology

3.1 Overview

This chapter transforms broad theory presented in Chapter 2 into a specific and tangible implementation. Any additional concepts addressed in further detail in this chapter are design decisions or novel software created by the author. The overall goal of this thesis is to automatically exploit FMV and produce control outputs. Section 3.2 explains the common architecture of AFFTRAC. Section 3.3 explains the particulars of the FMV processing, partial structural point cloud generation, and outlier rejection. Section 3.4 demonstrates how the point cloud was used in occlusion detection. Section 3.5 shows how the occlusion matrix was converted into a holding area density. It also shows how that holding area density was exploited for control output. Section 3.6 covers real-world specific considerations necessary for the test flight. Finally, Section 3.7 defines the performance metrics upon which the system was evaluated.

This thesis has two modalities of AFFTRAC: simulation and real-world flight test. While the modalities shared a common architecture, they differed in their method of execution. The realities of flight test drove the common architecture to be split into sub-groups and required multiple artificial data injects. Section 3.2 - 3.5 detail the specifics of the common architecture, and Section 3.6 revisits any necessary sections where real-world implementation differs.

3.2 Common Architecture and State Transitions

Two things are detailed in this section. The first is the common architecture. This was tested differently depending on whether AFFTRAC is being used in simulation or for flight test. The overview of this can be seen in *Figure 24* and is re-inserted here

from Section 1.5. The second is a high-level overview of the AFFTRAC software.

3.2.1 Common Architecture.

As stated in Section 1.5, there were two modalities of AFFTRAC. The first modality was simulation. The rough data flow into and out of the common architecture in simulation be seen on the top of *Figure 24*. The notional data flow associated with flight test can be seen on the bottom. However, the actual flow of data in flight test was more complicated.

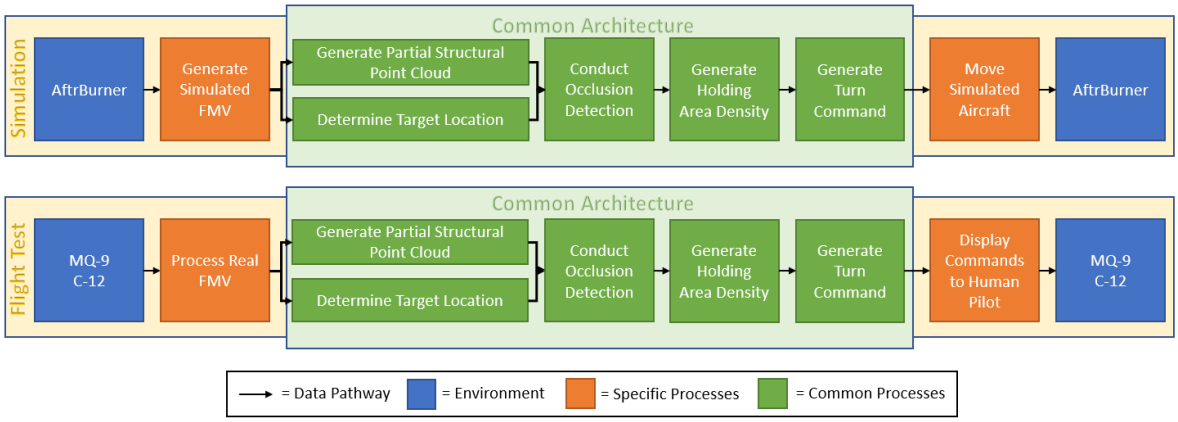

Figure 24: *Methodology Overview*

Whether in simulation or in flight test, the common architecture of AFFTRAC was as follows:

- Start with a full motion video source of a target environment.
- **Generate Partial Structural Point Cloud** - Using pairs of sequential images separated by sufficient parallax from aircraft motion during orbit, create a partial structural point cloud representative of the building environment.
- **Determine Target Location** - Using the same images as the previous step, localize the target in the image and place the target correctly in relation to the

partial structural point cloud.

- **Conduct Occlusion Detection** - Trace rays from the identified target (a person) against the partial structural point cloud and determine angles of visual occlusion.
- **Generate Holding Area Density** - Create a holding area composed of known good azimuths and radial ranges. The relative density of the holding area should drive the aircraft to stay at the radial mean and not favor any particular visible azimuth.
- **Generate Turn Command** - Given the current aircraft location and a holding area density, use a greedy algorithm (quadrant analysis) to produce turn commands that keeps the aircraft in the holding area density.
- Enact those turn commands in the current environment.

Simulation used simulated FMV and noiseless information. The aircraft could do whatever was commanded instantly, produced live video, and everything was synchronized and digitized. There were a number of major differences to consider when real data or real hardware was introduced. Namely, live video feed was not connected into the system, direct control of the aircraft was not permitted, and (due to lack of connectivity between navigation hardware and data collection hardware) location information had to be manually entered for both the aircraft and the target in relation to the structural point cloud. This was done not because of technical limitations but because of an inability to directly interface with the GCS. This was due to time, policy, and configuration management limitations.

In order to highlight the particular differences, first consider a block diagram of AFFTRAC in simulation in *Figure 25.* All the data were able to flow in real-time through the common architecture. The individual components are further detailed

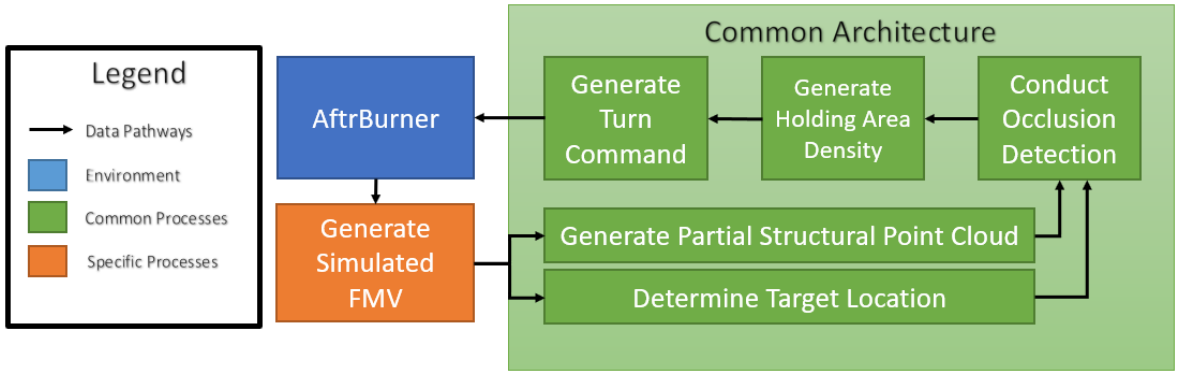

Figure 25: *AFFTRAC in Simulation*

Sections 3.3 - 3.5. However, as stated above, a direct interface between the test aircraft sensors and AFFTRAC was not available in flight test. This necessitated test-specific modifications to the system operation for each of the tested functions.

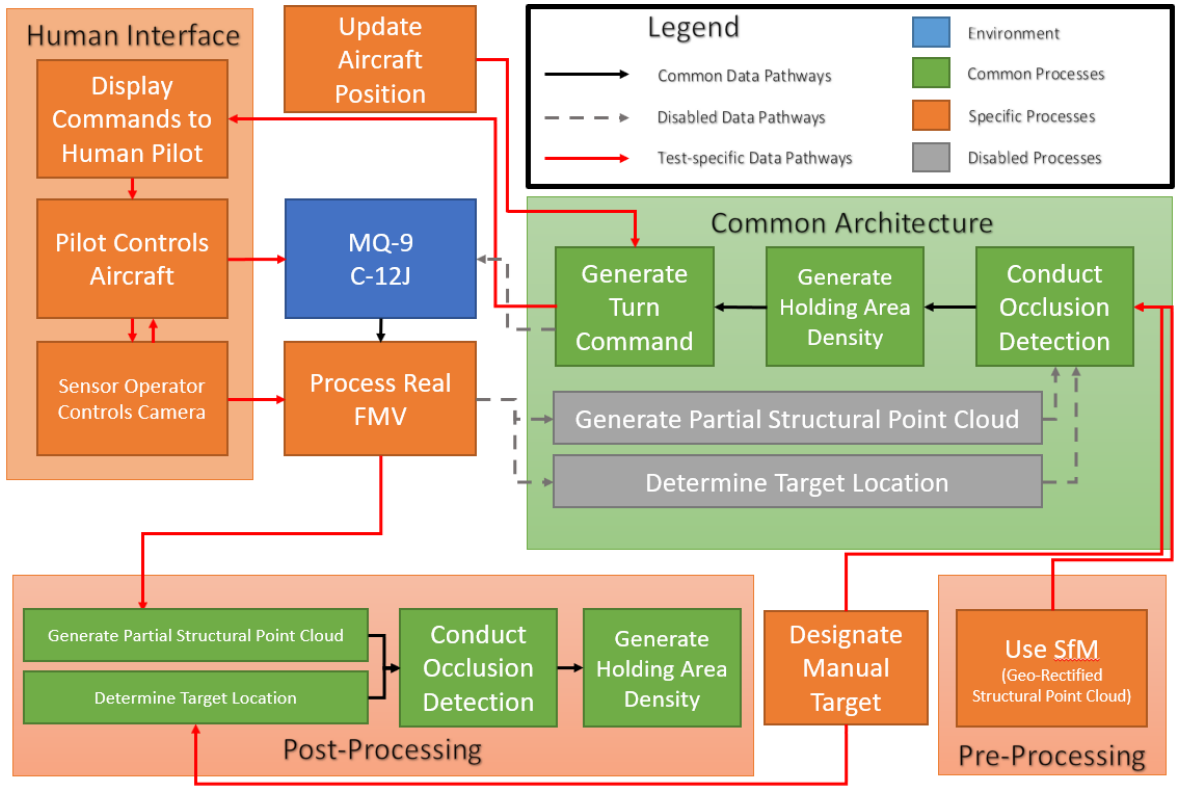

Figure 26: *Flight Test Specific Configuration*

Now consider the block diagram of AFFTRAC in its flight test configuration as seen in *Figure 26*. The grey dashed lines and process blocks in the block diagram represent the process flow if AFFTRAC was able to process video and telemetry from aircraft sensors in real-time and issue commands directly to the aircraft as it did in simulation. However, these data streams do not exist and have to be artificially replicated. The solid black lines represent data paths common to both simulation and the test configuration. The solid red lines represent the flight test data injects, which include any artificial data input to the system (i.e. it would not exist if AFFTRAC was operating with real-time data). There were multiple components necessary for AFFTRAC in flight test not necessary for simulation: the human interface, pre-processied SfM point clouds, designation of manual targets, manual update of aircraft position, post-processing of recorded flight video and telemetry.

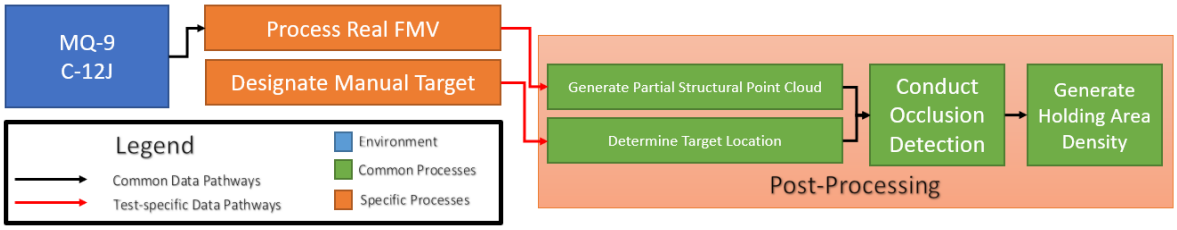

Figure 27: *Flight Test - Testing Holding Area Density Generation*

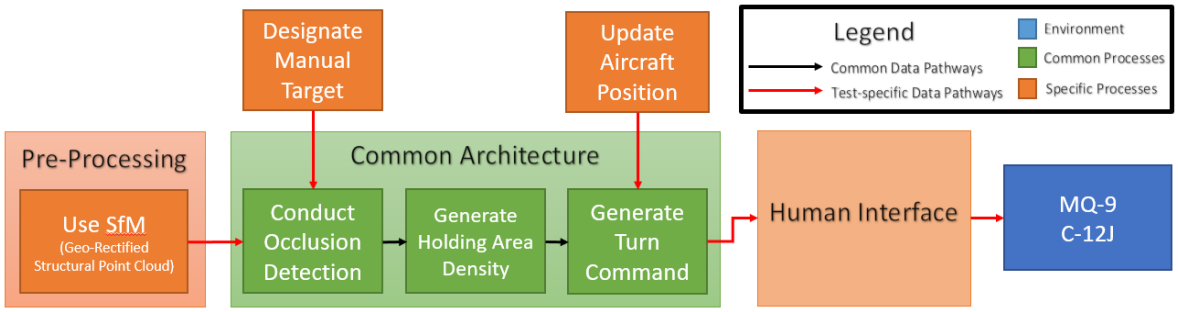

Figure 28: *Flight Test - Testing Aircraft Holding Performance*

While *Figure 26* shows how AFFTRAC was configured in flight test, it does not

show how it was specifically evaluated. The common architecture was split into three categories to allow as much evaluation of system-level integration as possible. The first category can be seen in *Figure 27*. This evaluated the generation of holding area densities from real imagery in post-processing. The next category evaluated only the generation of turn commands. The last category evaluated the portion of the common architecture as seen in *Figure 28*. This evaluated the holding performance of the aircraft. These components are detailed in Section 3.6 and Section 3.7.

3.2.2 AFFTRAC Hardware and Basic software function.

AFFTRAC was written in C++ using the AftrBurner [45] simulation software developed by Dr. Scott Nykl at the Air Force Institute of Technology. The AFFTRAC software was hosted on a standalone Lenovo P51 laptop computer equipped with an Intel i7 processor, 16 GB RAM and a Quadro M1200 graphics card. This computer was hand carried on board the C-12J and into the MQ-9 GCS during testing. The test article was a technology demonstrator and was not considered to be operationally representative.

The AFFTRAC software screen can be seen running in *Figure 29*. The main window was a free play window that can fly around the virtual world and take any camera vantage that is useful for analysis. This was also the window that was used to view the holding area density. This can be seen in later pages (*Figure 42* is the first example). The bottom right window shows what the camera, whether real or virtual, was currently seeing in terms of imagery. The bottom left window shows what AFFTRAC had reconstructed and understood about the ground environment. Not shown are the green points that make up the holding area density (reference *Figure 42* to see an example).

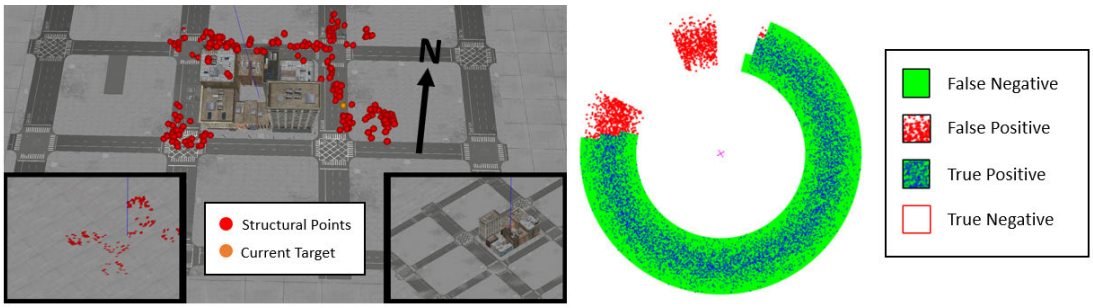

Figure 29: *AFFTRAC software screen - City Block*

3.3 Video Processing and Point Cloud Extraction

There were two distinct sources of video used in this thesis. The first was fully simulated and created in AftrBurner [45]. The second was actual aerial video and was either consumed from a playback file in post-processing or pre-processed prior to actual flight test. Section 3.3.1 describes simulation specific production of FMV. Real-world specific pre-processing and post-processing is covered in Section 3.6. Section 3.3.2 and 3.3.3 detail how image pairs were converted into a three dimensional point cloud known as a partial structural point cloud. Section 3.3.4 covers post-processing of the point cloud that was done to "clean" the point cloud prior to further use.

3.3.1 FMV Production and Pre-Processing.

The first step in exploiting simulated or real FMV is to calibrate the camera as stated in Section 2.5.4. This was accomplished by taking a series of images of an object with known geometric measurements in various orientations. AFFTRAC used a checkerboard. The images were convolved with an OpenCV convolutional filter to determine the location of the corners in the checkerboard. When a sufficient number of features were identified, the rotated checkerboard was matched to the subsequent points. Then, the known real distances between points are compared to the measured distances (Figure 30).

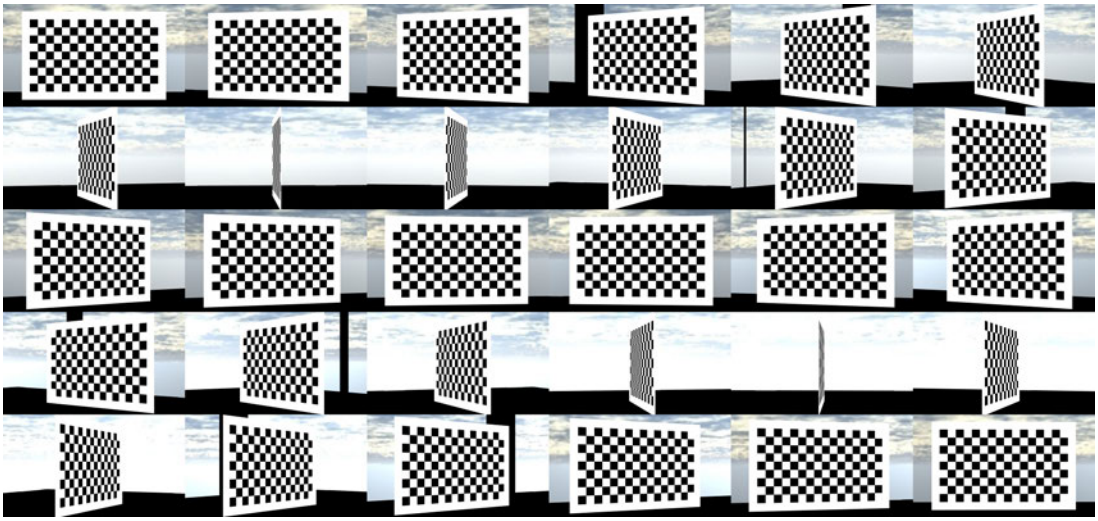

Figure 30: *Camera Calibration in Simulation*

Now, with a properly calibrated camera, footage was captured and exploited. In simulation, the FMV was produced artificially. This was done in the AftrBurner by taking a series of images. This can be visualized in *Figure 31*, though in execution the aircraft is not flying a pre-programmed path.

The images generated in simulation were too perfect. To correct this a long exposure Optical Transfer Function (OTF) was applied to the generated image. The mathematical explanation of why this effective and the exact method of derivation is outside the scope of this paper [25]. However, it can be understood to roughly model two things:

1. Noise introduced by atmospheric refraction during transmission.

2. Electromagnetic destructive/constructive interference as a result of restricting the EM propagation through a lens and pupil.

The OTF algorithm can be understood basically as follows:

- Transform the image to the Fourier domain

Figure 31: *Generating Images for Exploitation in Simulation - City Block*

- Create a low-pass Gaussian filter based on imaging system spatial dimensions and location
- Multiply the two together
- Transform the image back to the spatial domain.

To further simplify, as multiplication in the Fourier domain is mathematically equivalent to convolution in the spatial domain, the approximation becomes a convolutional mask as described in Chapter 2:

$$GaussianMask = 1/256 \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{17}$$

When this mask was convolved with the produced image, the end result is a loss in high-frequency information. The hard aliasing disappeared and the image slightly blurred. This roughly models the loss of information due to atmospheric interference and electromagnetic interference between photons when passing through the pupil.

Jumping ahead in processing the images, consider the end result of applying this OTF. *Figures 32* and *33* show what this decrease in feature count accomplishes. *Figure 32* had no Gaussian blur applied to the original images and the point count was much higher than *Figure 33* as a result. Additionally, it is worth noting that AFFTRAC did NOT produce a complete point cloud like this during its evaluation. It only produced partial structural point clouds as described in Section 3.3.3. These aggregated structural point clouds (they are in fact partial structural point clouds stitched together) were an artifact of early AFFTRAC development and are shown here only because they demonstrate structral point cloud density as a result of Gaussian image blur to model an OTF.

3.3.2 Image Pair Analysis.

SIFT is described in detail in Chapter 2. In this sub-section their use in AFFTRAC is shown to identify features in an image. In *Figures 34* and *35* the results can be seen on simulated images. The first image does not have the long exposure OTF applied. The second image does. The blurring can be seen if areas of detail are compared

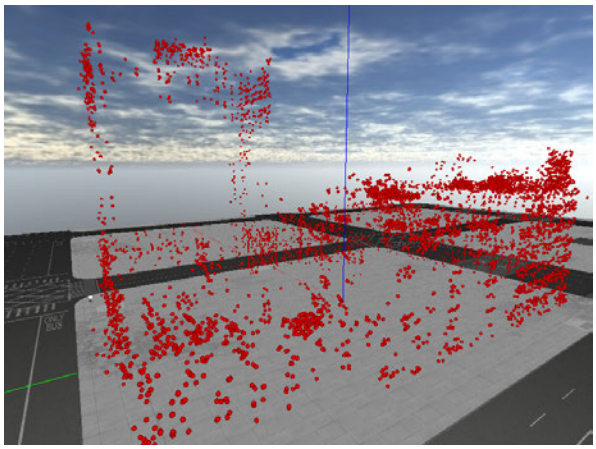

Figure 32: *Reconstructed Point Cloud with No Gaussian Blur - City Block*

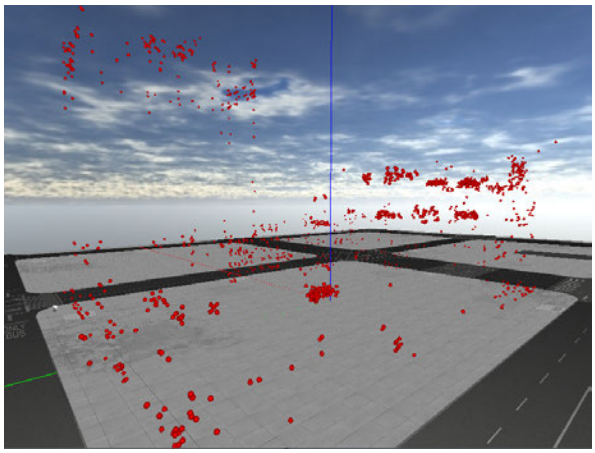

Figure 33: *Reconstructed Point Cloud with OTF Gaussian Blur - City Block*

between the two images. Additionally, the second image has far fewer features identified. The SIFT kernel is looking for distinct changes in intensity, which are lessened with the OTF as high frequency spatial information is lost by applying the OTF.

The next step in the image exploitation process was to do feature matching between subsequent images. *Figure 36* shows this process. Here a pair of sequential (in orbit) images separated by sufficient parallax underwent feature matching and re-projection into three dimensional space. This process was repeated for every sequential pair of images produced. In the current implementation, an image was

Figure 34: *Features Identified with No Gaussian Blur - City Block*

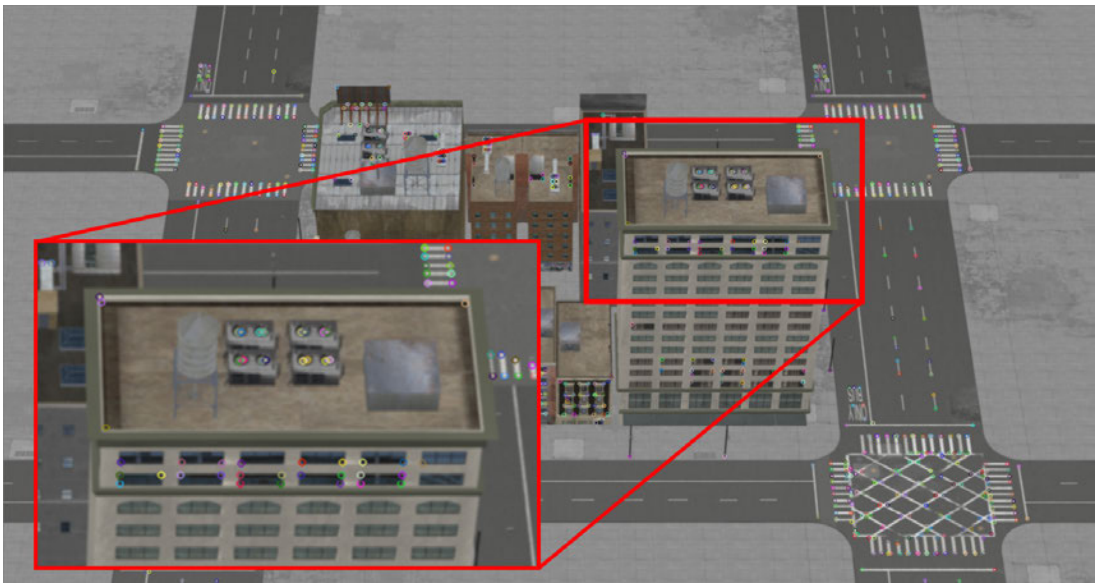

Figure 35: *Features Identified with OTF Gaussian Blur - City Block*

taken every 10 degrees of parallax. These points and known camera poses were then processed to determine rough spatial location. This is further explored in the next sub-section.

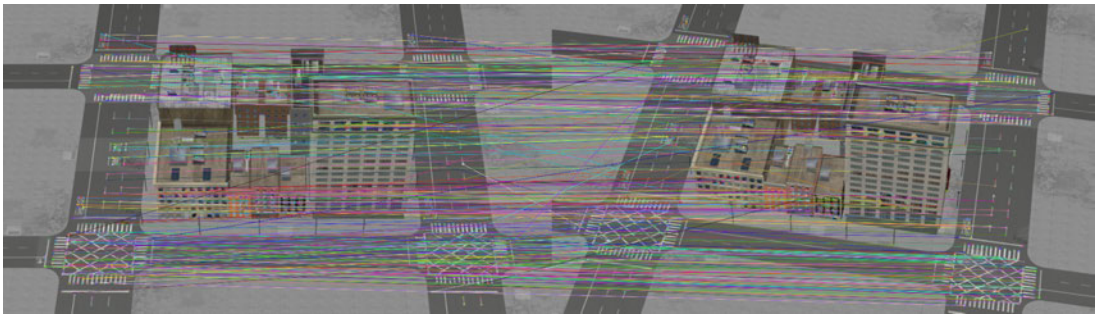

Figure 36: *SIFT Feature Matching between subsequent images - City Block*

3.3.3 Partial Structural Point Clouds.

AFFTRAC was designed to run in real-time. Thus, it needed to try to consider as small of a chunk of time as possible while still deriving the spatial information it required to operate. The smallest time step that can be used to recover three dimensional information from two dimensional images is a pair of images separated by a sufficient amount of parallax to accurately re-project the information into a partial structural point cloud. This was what AFFTRAC did when working in simulation as well as during flight test playback as detailed in Section 3.6. As mentioned in Section 2.5.3, the z dimension still needed to be determined. Section 3.3.2 produced a series of (x, y) features and associated coordinates in a plane perpendicular to the camera look direction. These features were matched between these (x, y) planes. Furthermore, the camera location and look location was known (from software in simulated and GPS in reality) in full (x, y, z) coordinates. Thus, the coordinates were projected through epipolar geometry back into three dimensional space.

The algorithm used for point cloud projection was developed by Arnold [5] and the code was heavily modified from his initial effort. The rest of this section goes into more detail in line with this overview. The basic overview is:

- Do feature matching between a pair of images with a sufficient amount of parallax between then (10 degrees)

- Identify which features actually match in spatial proximity to each other (i.e. features should be close to other like features in both images).
- Determine camera locations and orientation and pass those parameters into a stereo vision algorithm.
- Take the output points and triangulate their location with respect to the original camera location.
- Rotate the points back into a common world frame so all the points geo-locate correctly.

Feature matching starts with a pair of simulated images having the long exposure OTF kernel convolved over them. Each image had features and descriptors created as described earlier. Next, a Flann Based Matcher [5] processed the descriptors and determined what features were actually matches of each other. Afterwards, they were filtered through a K-Nearest Neighbor algorithm to make sure descriptors in both images were local to the same neighboring descriptors. The end result of the matching process was a a series of matched descriptors describing highly localized features. However, the matches still needed to be further processed by creating an essential matrix E. Here two sets of points (p and $q^{'}$) needed to be projected into 3D space. Each 2D point had an x and y location and a know focal length f. However, they existed in different coordinate reference systems. Solving the relationship between the points in the coordinate system is the basis of the essential matrix E and is defined in [70]

$$p = \begin{bmatrix} p_1 \\ p_2 \\ f \end{bmatrix} \qquad q^{'} = \begin{bmatrix} q_1^{'} \\ q_2^{'} \\ f^{'} \end{bmatrix} \tag{18}$$

Points can be translated into each other's reference system as follows (where R is a rotation matrix and t is a translation vector):

$$p^{'} = R(p - t) \tag{19}$$

Putting these epipolar lines into matrix format defines $[t]_x$

$$[t]_x = \begin{bmatrix} 0 & t_3 & -t_2 \\ -t_3 & 0 & t_1 \\ t_2 & -t_1 & 0 \end{bmatrix} \tag{20}$$

Combining the rotating matrix creates the essential matrix E

$$E = R[t]_x \tag{21}$$

This allows a compact relationship to be expressed between p and $q^{'}$

$$(q^{'})^T Ep = 0 \tag{22}$$

After this relationship was known, the matched pairs of points were then again individually considered to see how well they fit this known transform. Points that had a low probability of having their locations in the two images adequately described by the E were rejected. The points were now ready to be passed to a stereo rectify algorithm (OpenCV). This algorithm took in a K-matrix from the camera, a rotation matrix R that represented the relative rotation between the two camera poses, and a translation vector T represented the relative translation between the two camera locations. The output was a Q matrix representing a disparity to depth mapping, two separate rotation matrices ($R1$ and $R2$) from the midpoint, two separate translation vectors ($T1$ and $T2$) from the midpoint, and two separate projection matrices ($P1$

and $P2$).

$$Q1, R1, R2, T1, T2, P1, P2 = stereoRectify(K, R, T) \tag{23}$$

Before the points could be triangulated, they had to be rectified (OpenCV). The projection matrix, image locations of the features, K matrix, and rotation-from-midpoint matrices were combined to produce a set of rectified points for each camera. In the following equation, r represents a set of rectified points.

$$r_{cam1} = undistortPoints(K, R1, P1) \tag{24}$$

With the points rectified and undistorted, they could now be triangulated (OpenCV) into single set of points (ρ) in a common coordinate frame.

$$\rho = triangulate(r_{cam1}, r_{cam2}, P1, P2) \tag{25}$$

However, the points were still in an arbitrary coordinate frame and need to be translated into a common coordinate frame for subsequent display between different pairs of cameras. First, the points needed to be rotated so that the pose matches mid-point rotation between the two cameras used by OpenCV. Because OpenCV has different axis than AftrBurner the x and z axis needed to be swapped ($OpenCV_{rot}$). Additionally, the entire coordinate system needed to be rotated 180 degrees about the y axis (y_{rot}). Finally, the points need to be placed into the first camera's coordinate frame. The rotated end point e was derived as follows:

$$e_i = -cam1_{rot}^T \times y_{rot}^T \times OpenCV_{rot}^T \times R1^T \times \rho_i \tag{26}$$

Finally, the points needed to be translated back to the origin. The distance of both cameras from the focal point was known but the location of the midpoint between

them is not. Thus, the magnitude of the bisecting vector needed to be calculated so that the vector from camera 1 to the focal point can be scaled.

$$||T_{diff}|| = \sqrt{(T_{C1} - T_{C2}) \times (T_{C1} - T_{C2})^T} \tag{27}$$

$$||BiVec|| = \sqrt{\frac{||T_{C1}|| \times ||T_{C2}||}{||T_{C1}|| + ||T_{C2}||^2} \times ((||T_{C1}|| + ||T_{C2}||)^2 - ||T_{diff}||^2)} \tag{28}$$

$$f_i = e_i + (\frac{T_{C1}}{||T_{C1}||} \times ||BiVec||) \tag{29}$$

The result of this process was a partial structural point cloud (f) roughly representative of the original structure. *Figures 37* and *38* show this reconstruction in simulation. Additionally, the point clouds shown had already undergone outlier detection as detailed in Section 3.3.4. It is difficult to see in these figures, but there were no points that exist on the backside of the structure as viewed from this perspective. In other words, if the buildings were viewed from the opposite vantage, it would be as if that portion of the building did not exist. There was no spatial information retained from earlier image pair comparisons. This was by design. This allowed the most temporal ground environment to be captured. It also reduced computational cost.

But most importantly, during development, position and rotation errors were injected into this process to see their effect. This was done because this irreducible error was something expected to arise during real flight test (which it did as further described in Section 3.6 and Section 4.3). This produced an error in re-projection (several feet and several degrees of rotation) of the partial structural point clouds. It essentially became impossible to join the structures into a cohesive and complete structure. The end result is that AFFTRAC only produces partial structural point

clouds based on the current image taken and the previous image from 10 degrees of parallax prior to the current image.

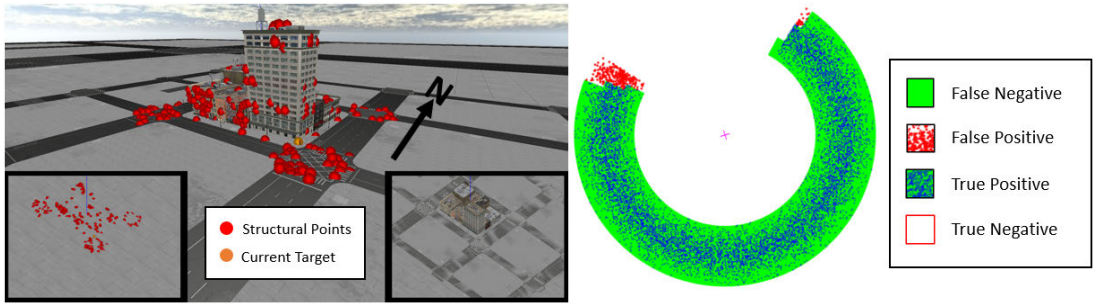

Figure 37: *Partial Structural Point Cloud - City Block*

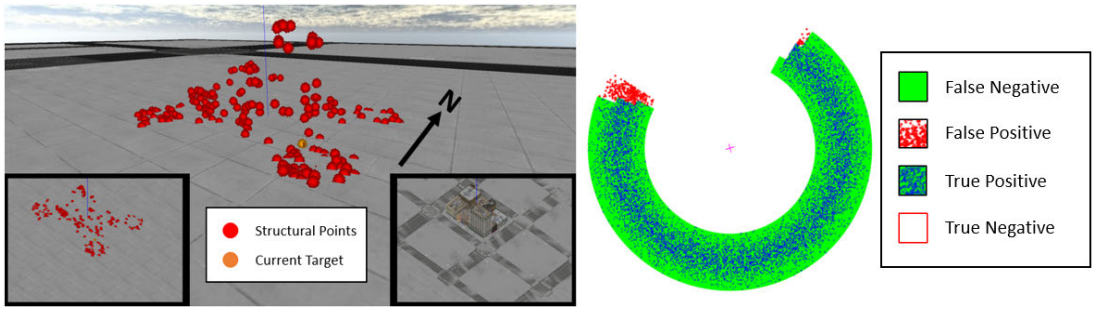

Figure 38: *Partial Structural Point Cloud with building hidden - City Block*

3.3.4 Point Cloud Post-Processing.

The point clouds generated from the previous algorithms are noisy. They have both transmission (non-structural) noise and surface (structural) noise [4]. Surface noise was ignored due to scale as the building only needs to be modeled approximately. Transmission noise can be seen in maroon in *Figure 39*. The structural points can be seen in blue. The desired goal was to clean the point clouds in such a way that they resemble *Figure 33*. Again, it is worth re-iterating that AFFTRAC did NOT produce a complete point cloud like this during its evaluation. It only produced partial structural point clouds as described in Section 3.3.3. These aggregated structural

point clouds (they are in fact partial structural point clouds stitched together) are an artifact of early development and are shown here only because they better show the complete structure and demonstrate outlier rejection.

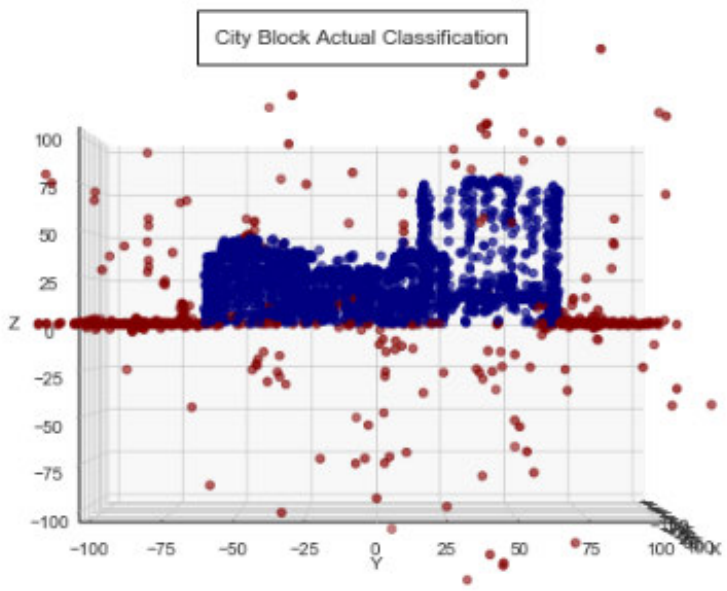

Figure 39: *Outliers in Generated Point Cloud - City Block*

There is a large body of research into cleaning point clouds. Exploring it is outside the scope of this paper. However, the author modeled various methods and arrived at a combination of methods to clean point clouds useful to this endeavor. Specifically, it was a sequential execution of two separate methods. The first was rejection of points outside the 90^{th} percentile radial distance from centroid. The second was a standard k-Nearest Neighbor (kNN) [30] that rejected everything outside of the 90^{th} percentile of the remainder. Radial distance was calculated for all N points p as follows:

$$r_m = \Sigma_{i=0}^{N}(\sqrt{{x_i}^2 + {y_i}^2 + {z_i}^2}) \tag{30}$$

All radial distances were then sorted and thresholded at the 90^{th} percentile. All points greater than the cutoff were removed from the point cloud. The subsequent

point cloud was then fed into a kNN algorithm with $K = 20$.

$$d_i = \Sigma_{k=0}^{K}(\sqrt{(x_d - x_i)^2 + (y_d - y_i)^2 + (z_d - z_i)^2}) \tag{31}$$

This algorithm found the distance d of every i point to its k nearest neighbors. The resultant aggregated distance d_i for every point was then sorted as in the radial distance. A threshold value was again selected at the 90^{th} percentile and points with cumulative nearest-k neighbor distances over this threshold are pruned. The end result of this process produced point clouds that are roughly representative of the original structure. This can be seen in *Figure 39*. The maroon dots represent the point identified as outliers and the blue dots represent what is identified as the actual structure. It can be seen that the structural point clouds maintained sufficient structural information for use in Section 3.4.

3.4 Occlusion Detection

3.4.1 Determining Target Location.

Positive identification of a target on the ground is a massive area of research by itself. Trying to solve this problem is outside the timeline and scope of this thesis. Therefore, a simplifying assumption was made. Specifically, the person (whether simulated or real) was wearing something to make them visually distinct. In normal color spectrum video they wore bright orange clothing. Currently, the target was inserted into AFFTRAC by moving around a human-like target. This target can be seen in *Figure 40*. In simulation, the location of this target was used within pixel space of the frame-by-frame image comparison to rectify the target location in relation to the structural point cloud.

By having a distinct shade of orange, a target was easily distinguished in the

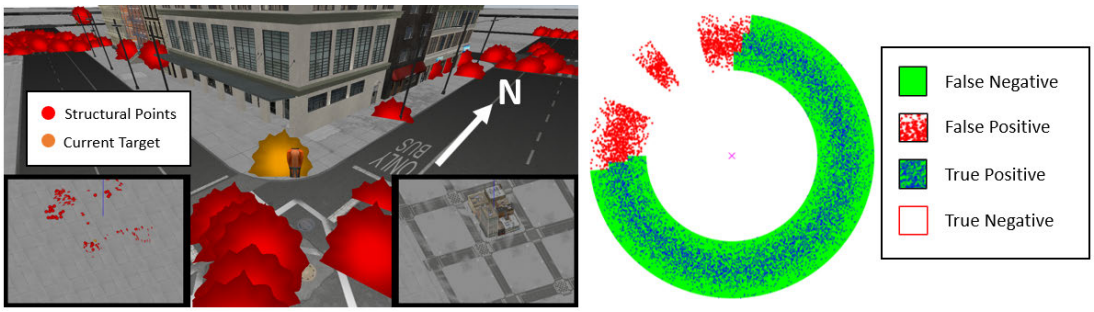

Figure 40: *Manually Inserted Human Target - City Block*

image. An image mask was generated by searching for a particular shade of orange. Once the target was localized in two dimensional space, it was projected into three dimensional space in exactly the same manner as outlined in Section 3.3. Any points generated within the image mask were separated from the partial structural point cloud. The points were then re-introduced back into simulation as a target point within the partial structural point cloud. The orange dots can be seen in *Figure 40* as well as *Figures 37 and 38* from the previous section. These orange dots show where the target has been identified and placed in relation to the partial structural point cloud.

3.4.2 Ray Casting and Occlusion Detection.

Once the target was determined as in 3.4.1, it was then used as a reference point to determine occlusion. This happened in the following manner: First, vectors of functionally infinite length (100 km) were drawn out from the target at increments of 10 degrees of ϕ and θ. All radial ground angles of θ were sampled every 10 degrees resulting in 36 separate radials. Only one quadrant of angles of ϕ were sampled. Within that, only angles between 30 and 60 degrees of inclination from the ground were sampled. This resulted in 4 angles of ϕ for every radial and a total of 144 separate vectors.

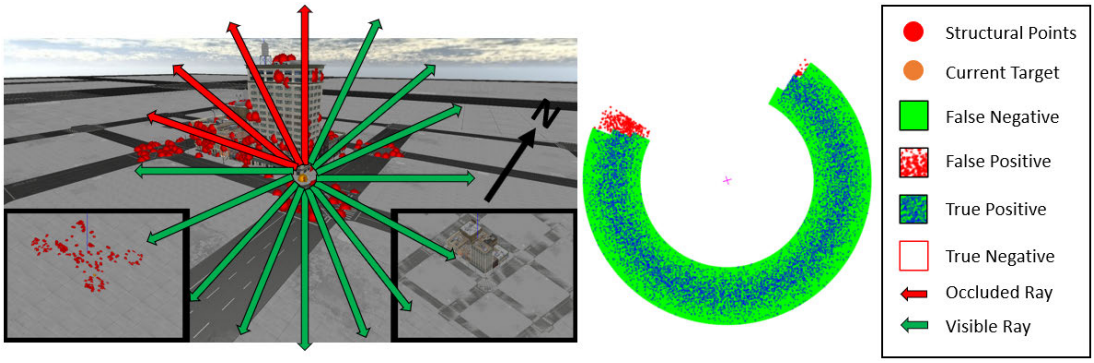

Figure 41: *Ray Casting Visualization - City Block*

Next, every point in the point cloud was translated into an array of vectors with the target as the origin. Each ray a was then compared to all i vectors b_i derived from the point clouds. The comparison takes the trigonometric form:

$$\theta = cos^{-1}(\frac{a \cdot b_i}{|a||b_i|}) \tag{32}$$

If theta was below 10 degrees, the ray was counted as "hitting" the point and considered "occluded". The search for that ray was then terminated and the ray was marked occluded. If all points in the point cloud were considered and the 10 degree threshold was never met, the ray was considered "visual". The end result of this process was an occlusion matrix for all rays of particular ϕ and θ. An example can be seen in *Table 1*

Table 1: *Abridged Occlusion Matrix Example*

		Inclination			
		30	40	50	60
Theta	000	X	X	X	X
	010	X	X		
	020				

⋮

Theta	330			X	X
	340			X	X
	350	X	X		
		30	40	50	60
		Inclination			

3.5 Holding Area Density Generation and Exploitation

This section describes how the information generated from previous sections is translated into actionable information. Specifically, it starts by detailing how the occlusion matrix is directly translated into a value map as described in Section 2.6. Next, the exact nature of the control algorithm is described. Note as well that this also represents the end of simulation specific architecture as mentioned at the start of the chapter. Section 3.6 addresses the flight test modality.

3.5.1 Generating a Holding Area Density.

AFFTRAC generated a static value map based on known heuristics. The composition of the value map was based on human judgement rather than ability to get to a high reward state through iterative analysis of position and movement [63]. As a Reaper's altitude generally remains constant, the z axis was eliminated with a constant altitude. The x and y coordinates were further restricted by minimal and maximal radial (r) distances as seen below:

$$R(x,y) = \begin{cases} 1, & \text{if } R(\theta,\phi)\ AND\ minDistance \leq r \leq maxDistance \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

A Gaussian distribution was used for the radial dimension:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{34}$$

This allowed for a heavy biasing of good location towards the mean of the distribution. If applied to *Equation 33*, the value for 1 is replaced by $P(r + minDis)$ as seen in *Equation 35*.

$$R(x,y) = \begin{cases} P(r + minDis), & \text{if } R(\theta,\phi)\ AND\ minDis \leq r \leq maxDis \\ 0, & \text{otherwise.} \end{cases} \tag{35}$$

The end result was a circular value map around a point. This distribution of the angular dimension was uniform unless restricted by the occlusion matrix described in the previous section. When restricted, the sampling of the remaining angles was uniform.

In *Equation 34*, μ is equal to 6 km and σ is equal to 0.75 km. These numbers were selected based on distances calculated in Section 2.4.3 to stay between 4.5 and 7.5 km. The bias towards the mean of the holding range was for image fidelity. Radial distance was calculated as ground distance to the target and not slant range. The initial value map was a wheel hold as the structure is unknown. The value map was generated by a Monte Carlo sampling of 100,000 n points. Azimuths were determined good or bad in 10-degree radials starting with radial 000. The points were generated

as follows for the initial wheel hold:

Algorithm 1 Value Map Generation

```
1: procedure GeneratePoint(r, θ)
2:     for i in n points do
3:         while (r < 4.5 || r > 7.5) do
4:             r ← randNormal(μ, σ)                    ▷ μ = 6, σ = 0.75
5:         end while
6:         while θ.occluded() do                       ▷ Occlusion Matrix
7:             θ ← randUniform(0, 35)                  ▷ returns integer
8:         end while
9:         probabilityMap[i] ← convertCartersian(μ, σ)
10:    end for
11:    return probabilityMap
12: end procedure
```

When considering the algorithm above for a wheel hold, it was important to remember that no point cloud had been generated yet so all angles are not occluded. The end result can be seen in *Figure 42*. When the partial structural point cloud was completely generated and cleaned, this triggered the generation of a new holding area density. *Figure 43* shows a sectored holding area density where approximately 100 degrees (10 bad azimuths) of the hold orbit were considered occluded.

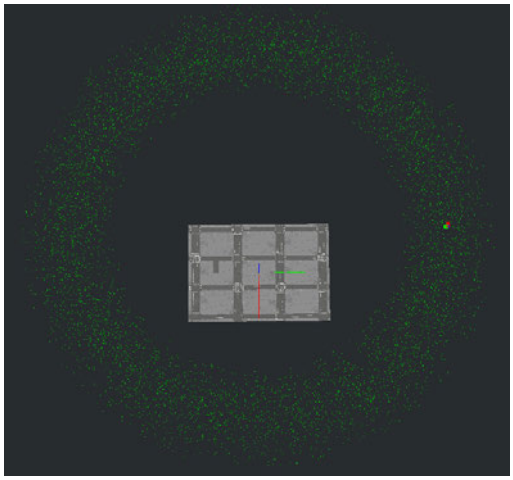

Figure 42: *Holding Area Density Example (Wheel Hold)*

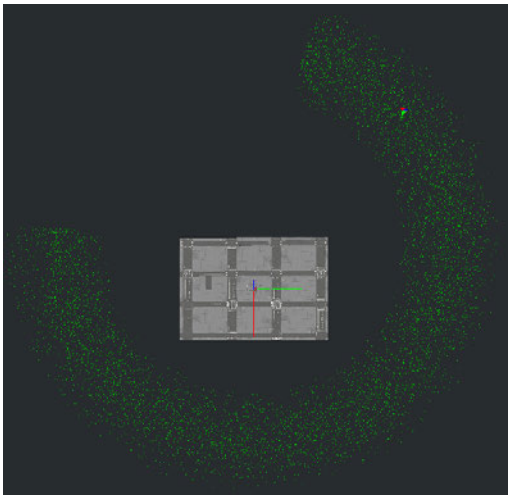

Figure 43: *Holding Area Density Example (Sector Hold)*

3.5.2 Dynamic Holding Area Density Behavior (Decay Rate).

An additional feature of the holding area density was that it decays over time. In fact, it decayed over 10 image pair comparisons. After the initial 100,000 points had been sampled to produce a wheel as described in Section 3.5.1, every subsequent image pair comparison produced 10,000 additional points. To keep the total at 100,000, points were deleted from a first in first out (FIFO) queue. This data structure meant that every 10 image pair comparisons had completely refreshed and decayed out all previous points. This capability was important as it allowed the aircraft to do metered seeking behavior. That is, if it lost the target for one time step, it had "memory" about where a good holding area used to be and would have most likely returned to a former good position rather than just assuming a wheel hold until the target was found again.

3.5.3 Aircraft Turn Command - Quadrant Analysis.

Regardless of how the holding area density was generated, the exploitation was the same. This could potentially give it an edge over conventional path planning

algorithms as it was agnostic to phase of flight or step of F2T execution. Additionally, the computational cost was minimal and entirely dependent upon the sampling frequency selected in the previous section. In its simplest form, it was fundamentally a quadrant analysis coupled with a greedy algorithm. The end result was a policy for the agent at the current time step based on the holding area density (or value map as in Section 2.6). A more formal definition can be seen below in *Algorithm 2*.

Essentially the aircraft had four quadrants relative to its current orientation. Front left, front right, back left, and back right. The quadrants extended out to a radius of 2.5 km from the aircraft. For each control time step, the number of points was aggregated in each quadrant. The totals were then compared to drive basic aircraft behavior: continue at current attitude, bank left, or bank right. In *Figure 44* two aircraft positions and relative quadrants have been drawn over simulation output. This provides a rough analogue of what happened in code. When all of these temporal greedy quadrant analyses were done in sequence, the end result was a path very similar to what was described in Chapter 2. The end result can be seen roughly approximated in *Figure 45*.

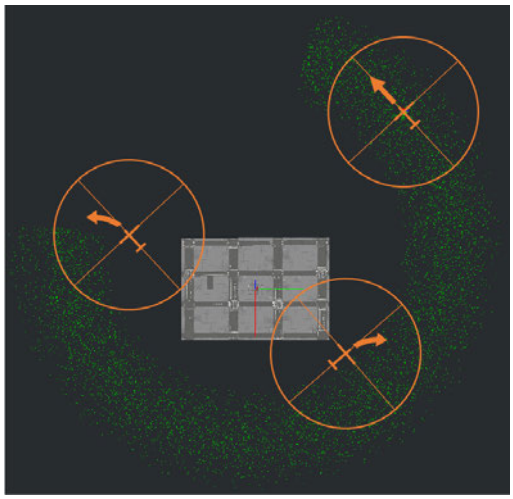

Figure 44: *Sector Hold Quadrant Analysis*

Algorithm 2 Greedy Decision Heuristic

```
1: procedure QuadrantAnalysis(aircraftLoc, probabilityMap)
2:     for i in n points do
3:         r ← dist(aircraftLoc, probabilityMap[i]
4:         if (r < 2.5) then
5:             quadTotals++ ← whichQuadrant(aircraftLoc, probabilityMap[i])
6:         end if                          ▷ Angle/Quadrants are from aircraft nose
7:         if quadTotals[allQuadrants] == 0 then
8:             quadTotals++ ← findNearestPoint(aircraftLoc, probabilityMap)
9:         end if
10:    end for
11:    if quadTotals[frontTwo] × 2 > quadTotals[allQuads] then
12:        if quadTotals[frontLeft] > quadTotals[frontRight] then
13:            if quadTotals[backTwo] = 0 then
14:                turnDir ←′ left′                          ▷ turnRate = 3°/sec
15:            else if quadTotals[frontRight]/quadTotals[frontTwo] < 0.25 then
16:                turnDir ←′ left′                          ▷ turnRate = 3°/sec
17:            else
18:                turnDir ←′ straight′
19:            end if
20:        else
21:            if quadTotals[backTwo] = 0 then
22:                turnDir ←′ right′                         ▷ turnRate = 3°/sec
23:            else if quadTotals[frontLeft]/quadTotals[frontTwo] < 0.25 then
24:                turnDir ←′ right′                         ▷ turnRate = 3°/sec
25:            else
26:                turnDir ←′ straight′
27:            end if
28:        end if
29:    else
30:        if quadTotals[leftTwo] > quadTotals[rightTwo] then
31:            turnDir ←′ left′                              ▷ turnRate = 3°/sec
32:        else
33:            turnDir ←′ right′                             ▷ turnRate = 3°/sec
34:        end if
35:    end if
36:    return turnDir
37: end procedure
```

3.5.4 Aircraft Motion model.

AFFTRAC was capable of a fairly complicated point mass motion model. It modeled the forward motion, current turn rate, and yaw rate irrespective of computational

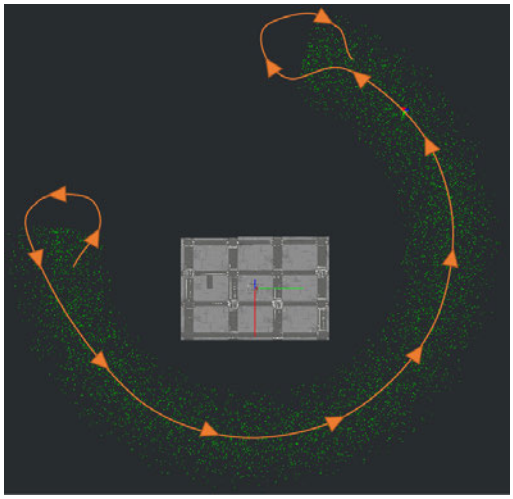

Figure 45: *Emergent Sector Hold Path*

speed. It used a high quality software timer to ensure that the physics was accurate down to the nano-second. It took parameters for control interval, time to standard rate turn, and control delay. Additionally, there was a rudimentary wind model that assumed constant winds everywhere in the simulation. All of the capabilities were used in flight test.

All simulation work assumed that the aircraft was flying at 140 KTAS as that is roughly representative the max endurance airspeed for the MQ-9 within the heart of its envelope. The control frequency for simulation was a turn allowed every 3 seconds. The turn rate from straight and level to standard rate turn took 3 seconds and the ramp up was linear. The current modeled delay was 0 seconds for simulation as well.

3.6 Flight Test Configuration

The AFFTRAC flight test implementation followed this rough chronological execution flow:

1. Collect 360-degree imagery of the target building (with no designated target) in a sortie prior to the current execution sortie.

2. Pre-process the collected imagery using SfM to produce a geo-rectified structural point cloud.

3. On the day of execution, load the geo-rectified structural point cloud into AFFTRAC.

4. Start AFFTRAC with the aircraft at a known position, heading, and airspeed.

5. Update the aircraft position in real time using a commercial GPS receiver (C-12J) or through manual updates (MQ-9), supplemented by an aircraft motion model between updates.

6. Manually inject the target location into AFFTRAC.

7. Observe AFFTRAC update the holding area density.

8. Observe the turning commands generated by AFFTRAC using quadrant analysis.

9. Manually fly the turn commands using standard rate turns (as required for a given test category).

10. Get the sortie data from the GCS racks and post-process to determine how AFFTRAC would have done in image pair analysis given the real-world scenario and data.

It is important to re-iterate that AFFTRAC in flight test had three distinct phases surrounding every sortie. The first was pre-processing of existing video data to produce SfM full structural point clouds that did not exist in any form during simulation. The second was active running of AFFTRAC in-flight using that pre-processed data. The third was post-processing data collected to see how AFFTRAC would have operated if it had live data feeds.

There were multiple components necessary for AFFTRAC in flight test not necessary for simulation: the human interface, pre-processied SfM point clouds, designation of manual targets, manual update of aircraft position, post-processing of recorded flight video and telemetry. These can be seen in *Figure 26* in Section 3.2. The remainder of this section details specifics about these five areas. Specifics about how the data is processed can be referenced in *Appendix D*.

3.6.1 Human Interface.

AFFTRAC as installed in an MQ-9 GCS can be seen in *Figure 46* and *Figure 47*. It consists of an AFFTRAC operator display and a turn direction arrow displayed to the pilot. The only hardware connection was a DVI cable from the AFFTRAC laptop to the monitor in the far top left of the GCS. This was used to extend the AFFTRAC laptop's desktop so that the turn indicator (*Figure 47*) could be seen and followed by the pilot (left seat). The main AFFTRAC display was on the AFFTRAC laptop out of screen from as seen in *Figure 46*. There are six roles that were filled to actuate the flight test:

- **MQ-9 Pilot** - Was responsible for general safety and control of the aircraft. Flew pre-scripted routes for data analysis, flew a completely human-controlled run scenario as a baseline, and followed AFFTRAC commands to act as a human affector of AFFTRAC turn commands.
- **MQ-9 Sensor Operator** - Was responsible for general safety and control of the aircraft ANDAS-1 sensor ball. Worked to keep the image high quality (stable, focused, etc.) during AFFTRAC commanded runs or participated in conducting ISR during human controlled baseline scenarios.
- **Test Conductor** - Was responsible for deciding what tests should be run in

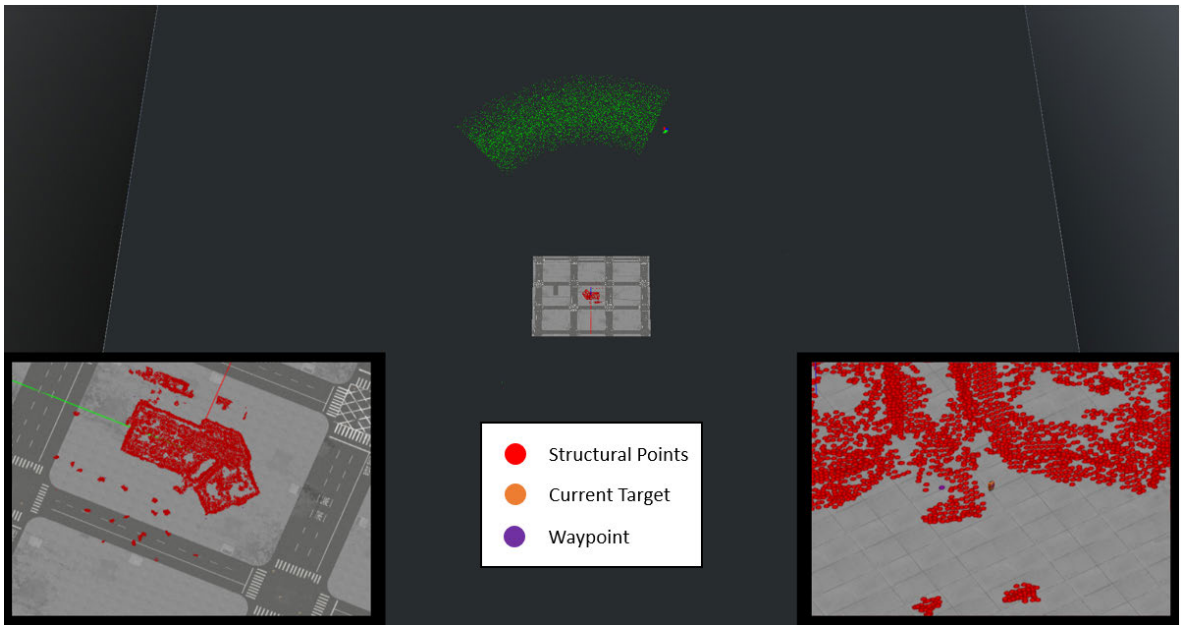

Figure 46: *AFFTRAC Flight Test Main Screen - Intermediate 1*

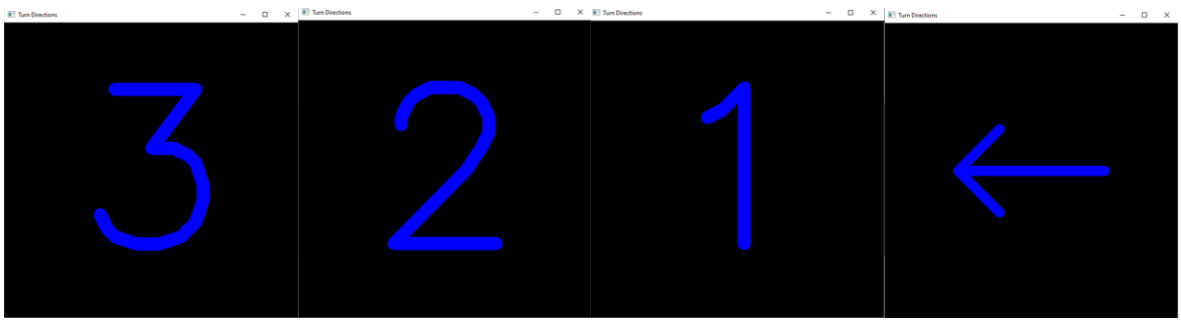

Figure 47: *Turn Command Countdown and Direction Arrow*

what order, when the test were conducted adequately and complete, and instructed other roles about their duties during test execution.

- **AFFTRAC Operator** - Was responsible for loading the correct scenario during appropriate test, moved the target to match real motion on the ground, and ensured that data were being recorded correctly in both the AFFTRAC laptop and the GCS video/telemetry recorders.

- **Coordinate Data Entry Operator** - Was the same person as the AFFTRAC Operator. Was responsible for reading location and heading coordinates from the GCS and typing them into the AFFTRAC Laptop GUI for coordinate entry.
- **Ground Site Crew** - Was responsible for setting up executing pre-scripted ground scenarios on-site at the request of the Test Conductor.

The display seen by the AFFTRAC operator during live test flight can be seen in *Figure 46*. As before, any point in red is a structural point. Any point in orange is where AFFTRAC thinks the target is in relation to the structural point cloud. A new addition is a purple point. These are "way-points" used to make sure that what the ground site crew and the AFFTRAC operator have the same target position. The usefulness of these points is shown well in *Figure 48* where the purple way-points match the orange traffic cones on the ground. In fact, orange traffic cones were used in multiple scenarios for just this reason as well as to provide additional static targets for future exploitation of collected video/telemetry.

The main window was a "free-play" window where the operator could have looked at either the holding area density current state or zoom into the ground environment state and inspect more closely where the manual target, partial structural point cloud, or way-points are in the AFFTRAC understanding of the three dimensional space. The bottom left window shows a top down view at a height scaled to more easily show the purple way-points. The bottom right window shows the AFFTRAC structural point cloud from the aircraft sensors current location and perspective.

In order to run different scenarios, and internal AFFTRAC configuration file (aftr.conf) had to be altered and AFFTRAC had to be closed and re-started. At the beginning of each run, the AFFTRAC virtual aircraft was placed in known position, heading, and airspeed that matched the starting parameters. When the actual aircraft was at the same location, the scenario was told to play and appropriate

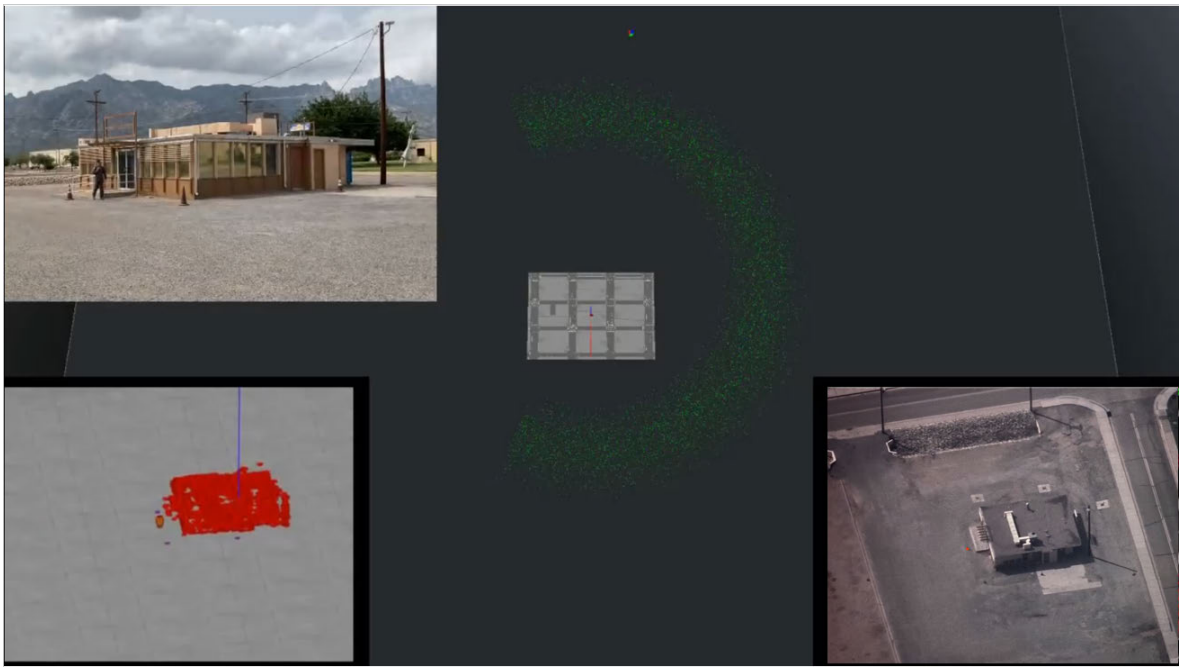

Figure 48: *Flight Test - Playback Visualization - Simple 1*

actions were taken to get necessary data.

3.6.2 Pre-Processed Structure from Motion Point Clouds.

As stated earlier, AFFTRAC was not capable of live video processing in flight test. Thus, an intermediary was necessary to create something representative of the structure prior to flight test execution. This is because, in order to have AFFTRAC issue commands for a human pilot to follow, there needed to be a holding area density for the aircraft to analyze. This holding area density was created from an occlusion matrix. This was in turn derived from the target location in relation to a partial structural point cloud. Without live video, there was no way to produce a partial structural point cloud. So a structural point cloud needed to be created prior to flight test and loaded into AFFTRAC.

It was decided to use SfM to create this structural point cloud. The reasoning

behind this decision was that it created the best possible three dimensional representation. These structural point clouds were not partial (i.e. had points on all sides of the structure) and of much higher quality and density than anything AFFTRAC produced. This produced an artificiality where the holding area densities were better than AFFTRAC would ever produce. This was done intentionally to produce definite separation of the real-time "Generation of Holding Area Densities" sections from the "Generation of Turn Commands" and "Holding Performance of the Aircraft". The latter two sections are thus evaluated assuming the first was functionally ideal. This way each section was evaluated independently.

Figure 49: *Original Building - Intermediate 1*

An example of a high-fidelity model is depicted in *Figure 50*. This high-fidelity model was produced using the Regard3D SfM algorithm [2] with full 360-degree imagery from the MQ-9 ANDAS-1 acquired months prior to use in flight test. An image of the original building can be seen in *Figure 49*

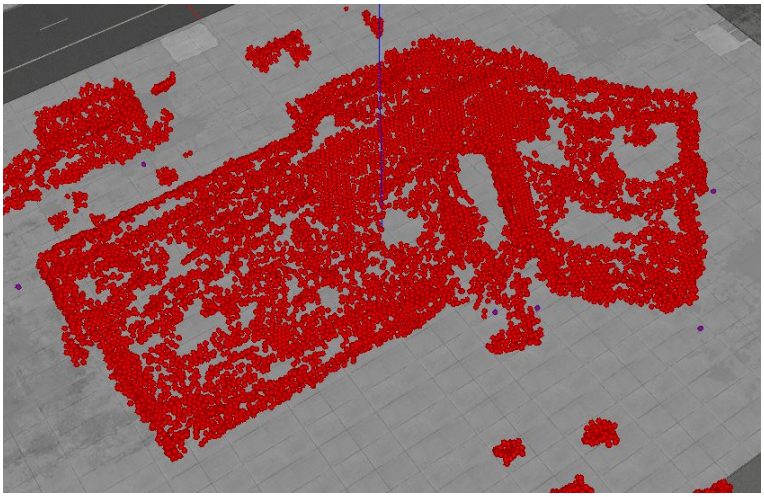

Figure 50: *SfM Re-creation - Intermediate 1*

3.6.3 Designation of Manual Targets.

In live flight test, target location updates were performed in real-time by manually moving a virtual target within the software representation of the environment, as depicted in *Figure 51*. This movement was coordinated with scripted movement of an actor on the ground. The common way-points can be seen with purple dots for virtual and actual target co-location. The target could be moved manually by the AFFTRAC operator in two ways. The primary method was to press a key to cause the target to automatically move to the next way point at 1.2 m/s. A backup method was to manually drive the target with arrow keys on the keyboard.

Video was recorded using the Multi-Spectral Targeting System (MTS) on the MQ-9, which showed target movement in the real world. This video was used in post-flight analysis. To help AFFTRAC accurately determine the targets location in post-processing, the target was overlaid with orange pixels, thus creating the visually distinct target AFFTRACs simple tracking capability relied on. This inject was used to test the occlusion detection and holding area density generation functions of AFFTRAC.

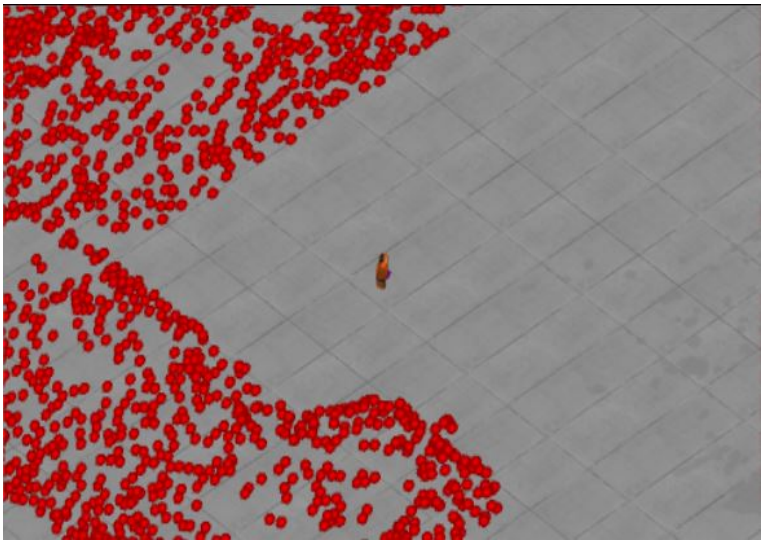

Figure 51: *Manually Controlled Human Target - Intermediate 2*

3.6.4 Manual Update of Aircraft Position.

None of the aircraft platforms provided a real-time position update feed to AFFTRAC. This was overcome in the C-12J through use of a COTS GPS puck, a GlobalSat BU-353-S4. The puck was a universal serial bus (USB) GPS receiver that featured a sensitive, low power consumption chipset in a compact form factor. This instrumentation was hand-carried onboard the C-12J during testing.

For the MQ-9, where the AFFTRAC computer was geographically separated from the aircraft, the test team manually entered MGRS coordinates and aircraft heading periodically. The MGRS coordinates were pre-filled so that the operator only had to type two sets of three numbers. This meant the location update was accurate to 100 meters. A motion model as described in Section 3.5.4 which assumed constant airspeed and standard rate turns augmented the manual location updates, increasing location fidelity. This inject was used to test all component functions of the AFFTRAC algorithm.

3.6.5 Post-Processing of Video and Telemetry.

As stated earlier, real footage was captured through the physical process of flying an airplane and using the video produced from the sortie. While MQ-9 video is digital and could be consumed by a third-party software such as AFFTRAC, the GCS was not configured to do so in flight test. Thus, collected video had to be post-processed after the sortie. An example can be seen in *Figure 52*. With sufficient difference in azimuth between images, i.e. parallax, AFFTRAC found common structural features, analyzed how they changed in the image, and produced points which represented features of the target environment.

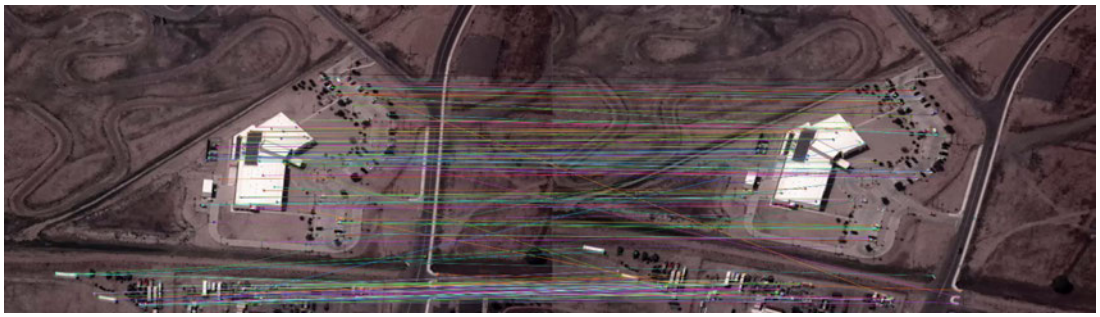

Figure 52: *Features Identified in Real FMV*

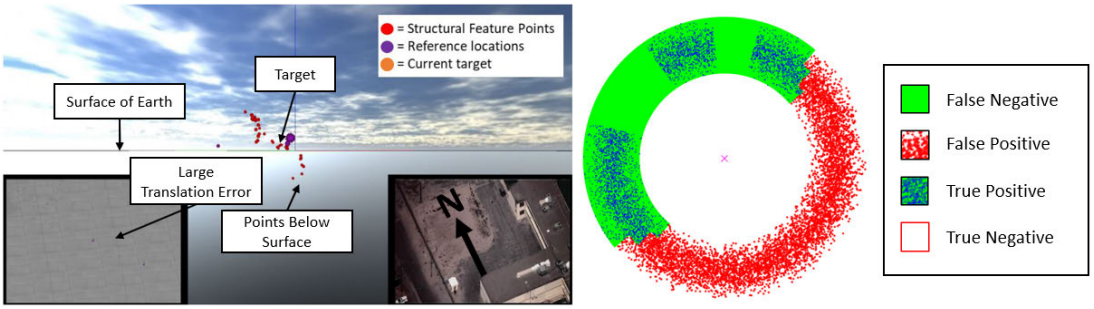

Figure 53: *Reconstructed Point Cloud in Real FMV - Intermediate 2*

Due to AFFTRAC's low technical maturity for flight test, significant post-processing was required to evaluate the system's occlusion detection and holding area density

generation performance. Following the flight test, collected video was processed in a simulated real-time environment and the output was compared to the true visual custody (determined by manual review of video) of a ground target during the same flight. From these simulated runs using flight test video AFFTRAC produced a series of partial structural point clouds, target locations, and resultant holding area densities. An example can be seen in *Figure 53*.

3.6.6 Aircraft specific considerations.

The MQ-9 motion model for flight test was slightly different than the model for simulation. In flight test, AFFTRAC assumed that the aircraft was flying at 180 KTAS. This was increased from 140 KTAS in simulation because the C-12 could not fly 140 KTAS without risk of stalling. The control input rate was increased to 7 seconds to allow time for human response time and command delay. The turn rate from straight and level to standard rate turn was set to 3 seconds and it was assumed that the ramp up was linear. This was not representative of the actual time to standard rate turn. Flight test showed it to be closer to 5 seconds. The control delay was modeled at 2 seconds.

The C-12J used in the test was intended to be fitted with a Sniper ATP produced by Lockheed Martin. The Sniper ATP is similar to the ANDAS-1 in that it is an EO/IR targeting pod used on an existing combat aircraft. However, due to maintenance problems with the auxiliary electrical system of the C-12J and the Sniper ATP control rack, it was unable to be used. Thus, the specifics are omitted. The C-12 did use a GPS puck as seen in *Figure 54* to pull live location information directly into the AFFTRAC laptop.

Figure 54: *GlobalSat BU-353S4 GPS Reciever*

3.7 Evaluation of Performance

This section defines performance measures for use in Chapter 4. It details three broad categories of evaluation (as first mentioned in Chapter 1) and describes the component metrics within. The two modalities that were evaluated are:

1. AFFTRAC performance in simulation
2. AFFTRAC performance in flight test

The three categories evaluated and specific metrics are shown below.

1. **Generation of Holding Area Densities**
   (a) Holding Area Density Score
   (b) Ability to Run in Real Time
2. **Generation of Turn Commands**
   (a) Correct Turn Command
3. **Holding Performance of the Aircraft**
   (a) Percent Time in Holding Area
   (b) Percent Time at Optimal Radius in Holding Area

(c) Percent Time Target Visual

These categories form broad categories to evaluate both simulation and flight test. This was done so that conclusions can be drawn about the performance of both in a common manner. Combining these modalities with processes evaluated resulted in a test matrix. This can be seen in *Table 2*.

Table 2: *Evaluation Matrix*

	Simulation	**Flight Test**
Generation of Holding Area Density	1a 1b	1a
Correct Turn Command		2a
Holding Performance of the Aircraft	3a 3b 3c	3a 3b 3c

Note that the "Correct Turn Command" was only evaluated during flight test. This was because evaluating whether the turn commands are correct is intentionally being considered independently of how the holding area densities were created. Additionally, "Ability to Run in Real Time" was only evaluated in simulation because there was no benefit in running it during flight test as simulation was more processor-intensive. This is because the partial structural points clouds were less dense in flight test and thus less computationally demanding to process.

Simulation performance serves as a baseline for how well AFFTRAC can perform in ideal conditions and was described in Sections 3.3-3.5. Any inclusion of real-world input or output introduces significant error so having a reference is valuable as benchmark. The flight test modality is described in detail Section 3.6 and is further detailed in Chapter 4. The main differences between the flight test and simulation

modalities are summarized below:

- The flight test configuration introduced considerable delay. Manual target coordinates had to be entered, the human pilot had to understand and actuate the command being given, and the aircraft took time to transmit, actuate, and achieve the desired angle of bank.
- There was actual sensor noise in terms of fidelity, aircraft position, sensor position and rotation, and update rate of telemetry
- The flight test had to be broken up into dis-jointed parts with numerous test injects as described previously

However, the MQ-9 was excellent at keeping a constant airspeed and altitude. Additionally, its turn behavior was consistent. This is because it can be commanded to actuate standard rate turns irrespective of the bank input by the pilot and achieves those turn in a regular manner.

3.7.1 Generation of Holding Area Densities.

The specific objective here was to evaluate generation of line-of-sight holding area densities in target-tracking scenarios. Runs began at a fixed radius in a circular holding pattern over a pre-planned building and ground target. The end of the run was when one 360-degree orbit was completed. The holding area densities were graded with a score designed to characterize how useful they are for maintaining visual line of sight. Also, the common architecture that created them was bench-marked to make sure that they can run in real time.

3.7.1.1 Holding Area Density Score.

Grading the holding area density was a functional way to determine how well the "Generate Partial Structural Point Cloud", "Determine Target Location", "Conduct Occlusion Detection", and "Generate Holding Density" portion of the common architecture as described in Section 3.2 (*Figure 27*) performed given the scope of this thesis. In other words, it graded the system-level integration of existing pieces of software with occlusion detection and holding area density generation.

The first measurement considered was a confusion matrix. For each of the holding area densities created in a run, each the AFFTRAC classification of each azimuth (occluded or visual) was compared against the visual area from manual review of video. This can be seen in *Figure 55*.

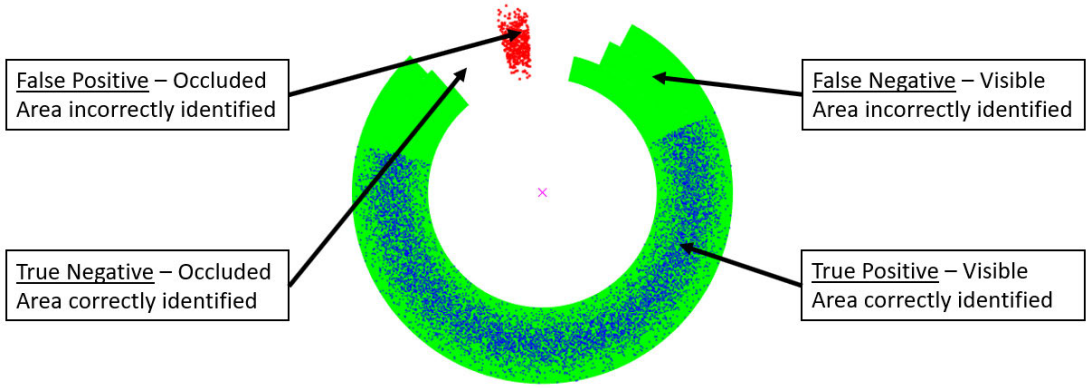

Figure 55: *Holding Area Density Confusion Matrix Visualization*

However, it was necessary to weight the different areas and normalize the results. Adding weighted risk was included because losing a target is a far worse outcome than not using the entire possible visual holding area for the hold. Normalizing the results was necessary because the addition of weighted risk. This allowed comparison between holding area densities with different size visual areas.

The result was a metric that scores each holding area density score made. This is

shown below with $\alpha = \frac{5}{6}$ across T total holding area densities with A azimuths per density. Here α is a risk weighting:

$$TotalPoints = \Sigma_{i=0}^{A} \begin{cases} \alpha, & \text{if } \theta_{predTrueNegative}[i] = \theta_{TrueNegative}[i] \\ 1-\alpha, & \text{if } \theta_{predTruePositive}[i] = \theta_{TruePositive}[i] \\ 0, & \text{otherwise} \end{cases} \tag{36}$$

$$NormalizedScore = \frac{TotalPoints}{(count(\theta_{negative}) \times \alpha) \ + \ (count(\theta_{positive}) \times (1-\alpha))} \tag{37}$$

An additional restraint placed on this metric was that only image pairs where the target was visible in both images were considered. The current logic of AFFTRAC simply creates full 360-degree holding area densities if the target was not found in the image pairs and localized in relation to the partial structural point cloud. As data were collected in a 360-degree orbit at a constant radius for this metric, it was not correct to evaluate what the algorithms produced when the camera is forced to go to a position where visual of the target is lost.

3.7.1.2 Ability to Run in Real Time.

The intent of this metric was to characterize how long it takes to run the core algorithms ("Generate Partial Structural Point Cloud", "Determine Target Location", "Conduct Occlusion Detection", and "Generate Holding Area Density") evaluated above. Additionally, outlier rejection was broken out from "Generate Partial Structural Point Cloud" into its own category in the results. "Generate Turn Command" was not timed because it is computationally cheap at $O(N)$. All of the other components are at a minimum $O(N^2)$. The timer started when both images to be analyzed were loaded into memory and stopped when the subsequent holding area density was

produced.

The common architecture as tested here only executed approximately every 10-15 seconds depending on airspeed. This was because there needed to be a sufficient physical distance traveled between image pairs to produce the necessary parallax. So at a bare minimum, the common architecture needed to execute in under 10 seconds. However, as that would have tied up all of the system resources, a more restrictive execution time of 200 ms was selected. This is a rough limit where 80% of the processor clock time is still available in any given second of processing.

3.7.2 Generation of Turn Commands.

This objective was to determine if aircraft turn commands would have maintained inside the holding area density or commanded a return to a holding area density. The run started at a fixed point in a holding area and flew a pre-determined ground track that goes in and out of the holding area. Data were collected every 5 seconds to determine whether generated turn commands were correct. A 360-degree holding area density was used and does not consider any partial or complete structural point cloud.

3.7.2.1 Correct Turn Command.

A MATLAB script was used to import the AFFTRAC logs and plot the circular holding area density, the actual aircraft ground track, and the AFFTRAC commands presented to the aircrew at the aircrafts location on the ground track. Specifics can be seen in *Appendix D*. Each of the aircraft commands was evaluated. A good command was defined as a command that would:

1. Maintain the aircraft inside a holding area density.

2. If outside, turn to return to the holding area density more directly than the other two options irrespective of any control delay.

3. Seek to orbit the target at the mean distance of 6.0 km within a standard deviation of 0.75 km.

3.7.3 Holding Performance of the Aircraft.

This objective was to evaluate the overall holding performance of AFFTRAC when its holding commands are followed in several scenarios. The characteristics of the aircraft flight path were analyzed quantitatively with three metrics. First was how much time the aircraft spends in the holding area density. Next, this was further refined to how much of that time is at an optimal radial distance. Finally, the flight path was analyzed to determine how much of the time the target was in visual line of sight of the target. In flight test, AFFTRAC's performance in these three metrics was also compared to the performance of a standard 360-degree orbit (where no attempt is made to keep the target in the field of view) and of a human instructor pilot (IP) (who attempts to keep a target in view with normal tactics).

Runs were conducted with both stationary and moving targets. Runs began at a fixed radius in a full 360-degree holding area density around the target building. For each scenario in flight test, a 360-degree constant radius wheel was flown around the target. Each run was 10 minutes of flight time. Each run began at the same initial conditions. In simulation, the entire common architecture was used when grading the holding performance of the aircraft. In flight test, only part of the common architecture (*Figure 28*) was used as described in Section 3.2.

3.7.3.1 Percent Time in Holding Area.

This metric measured how much of the time is the aircraft in green holding area density during the run. "totalTime" was the total time for the run. "timeInGreen" was the amount of time two constraints are simultaneously satisfied. The first constraint was that the aircraft is within the minimum and maximum radius of the holding area density. The second constraint was that there is a point in every quadrant of the aircraft (as described in Section 3.5.3) within 350 meters of the aircraft. This number was based on the minimum arc length (at a 4,500 meters radius) of a 10-degree empty section of 785 meters. Thus, if transiting between two disparate holding area densities at the minimum radius, there was never a point where being in an empty section would cause this metric to say that it is in the green holding area density.

$$percentTimeInGreen = \frac{timeInGreen}{totalTime} \tag{38}$$

3.7.3.2 Percent Time at Optimal Radius in Holding Area.

A more specific measure of hold quality is how much of the time was the aircraft at or near an optimal ground radial distance within the holding area density. This was defined as how much of the time is the aircraft within 1 SD of the Gaussian mean (μ is equal to 6 km and SD is equal to 0.75 km) of the holding area density. This was an additional constraint that must be simultaneously satisfied to the constraints described in Section 3.7.3.1 above.

$$percentOptimal = \frac{timeInOptimal}{totalTime} \tag{39}$$

3.7.3.3 Percent Time Target Visual.

This final metric measured how much time the aircraft was in visual line of sight of the ground target. In both simulation and flight test, this was determined by manually reviewing video. If any part of the target remained visible in the sensor field of view, visual line of sight of the target was said to have been maintained.

This chapter has covered the common architecture to AFFTRAC in both simulation and flight test. Within the common architecture, image pairs were compared to produce a partial structural point cloud with a relative target. That partial structural point cloud was used to produce an occlusion matrix which was used to produce a holding area density. Quadrant analysis exploited that holding area density to create emergent flight paths to maintain visual line of sight to a ground target. The next chapter shows the results of simulation and flight test data fed through the common architecture.

IV. Results

This chapter is divided into three sections. The first two sections (4.1 and 4.2) outline actual execution particulars for simulation and flight test. This includes what was actually flown, where it was done, what the scenario was, and what buildings were used in these scenarios. They also restate some important assumptions particular to either simulation or to flight test. The last section (4.3) covers the actual results broken out by the particular test category. Each test category was evaluated with the metrics described in Section 3.7. All of the metrics are constructed so that higher percentages are an indication of good performance.

4.1 Simulation Particulars

This section describes the particulars of AFFTRAC in simulation with actual execution specifics. Simulation creates a baseline of performance and shows how well AFFTRAC could work under near-perfect conditions. This section details the three test environments used and describes the sortie profiles. The main AFFTRAC window (as in *Figure 56*) in each image is positioned to best show each individual building and is not representative of simulated FMV that was actually produced. The FMV used in image pair analysis is in the bottom right portion of each figure. The partial structural point cloud produced as seen from the current aircraft position is in the bottom left of each figure.

4.1.1 Test Environments.

4.1.1.1 Notre Dame Cathedral.

The first test environment was a recreation of the Notre Dame cathedral. It was feature rich, large in relation to the target, and fairly simple in overall shape. The

spires of the cathedral faced due South and the building is oriented North/South. The test environment can be seen in *Figure 56*

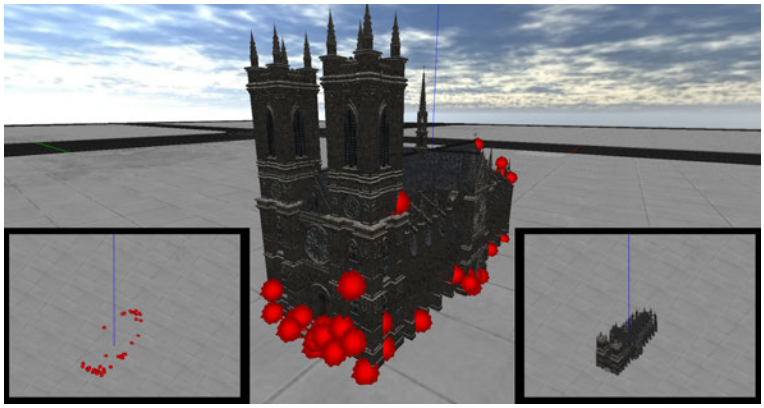

Figure 56: *Simulated Test Environment - Notre Dame Cathedral*

4.1.1.2 City Block.

The second test environment was a fairly complicated city block. It had some of the same desirable characteristics as Notre Dame. It was feature rich and large in scale in relation to the target. However, it was more complicated as it was comprised of several sub-structures. Additionally, it did not have a basic overall shape and there are strong ground features (the crosswalks) to pick up on as well. The test environment can be seen in *Figure 57*.

4.1.1.3 Single House.

This test environment, although smaller in scale and simpler than the first two, was more complicated to process due to lack of good features. Additionally, the smaller scale in relation to the target size actually made it more difficult to determine good occlusion angles. The test environment can be seen in *Figure 58*.

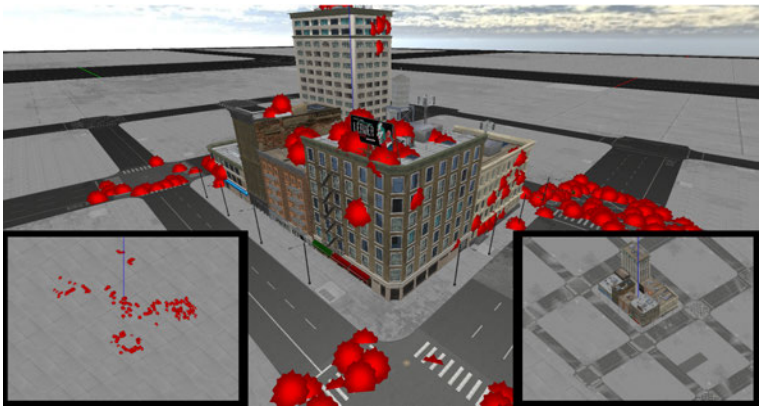

Figure 57: *Simulated Test Environment - City Block*

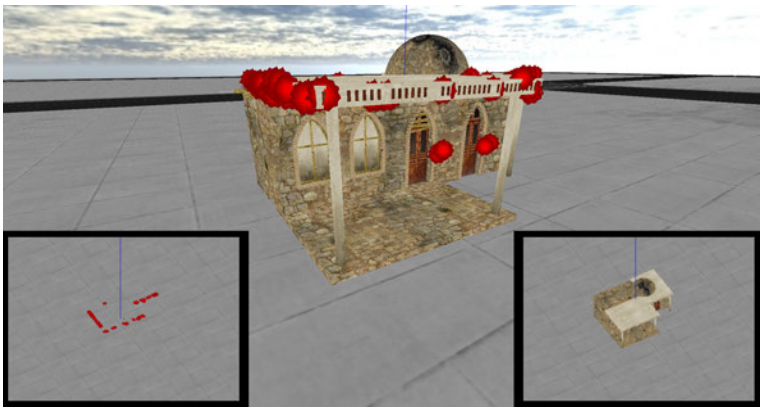

Figure 58: *Simulated Test Environment - Single House*

4.1.2 Sortie Profiles.

Each sortie begins with the virtual aircraft 6 km away from the target due east with the aircraft facing due north. The aircraft flew at 140 KTAS or approximately 105 knots calibrated airspeed (KCAS). Orbits were flown at 20,000 feet HAT. Once AFFTRAC was unfrozen, the run time started and the evaluation logger started storing data. The log file contained the following information:

- Timestamp
- Event flags (new holding area density created or new command issued)

- Current aircraft position and heading
- Current target position
- Current turn command
- What hold radials are denoted as good or bad

Additionally, in program analytics displayed necessary data for the percent time in holding area metrics during the run. The total run time of each run was 1 minute wall clock time. This translated into 10 minutes of "real-world" flight time as the simulation was running at 10x speed. Additional data reduction can be seen in *Appendix D.*

4.2 Flight Test Particulars

Flight test was conducted in the vicinity of the White Sands Missile Range (WSMR) at Holloman AFB, New Mexico from 8-18 September 2020 and comprised of four sorties totaling 8.6 flight test hours. Testing was split between two different types of aircraft. The first three sorties were flown on any available MQ-9 Reaper from the 49th Operations Group (OG) at Holloman AFB, NM. The last sortie was flown on a C-12J ("Mabel" tail number 86-0080), supplied by the 586th Flight Test Squadron (FLTS) at Holloman AFB, NM. The C-12J was only used to collect position information as it had no EO/IR sensor. AFFTRAC could not be evaluated as a whole system as in simulation. The nature of the flight test required that individual components be tested in isolation. For more details reference Section 3.6.

4.2.1 Test Environments.

Six different buildings were planned and mapped prior to test execution as seen in *Table 3.* Their location in the WSMR can be seen in *Figure 59.* All six buildings

Figure 59: *White Sands Missile Range*

were fed through SfM algorithms to produce high-quality geo-rectified point clouds. Additionally, a series of way-points were loaded around the SfM points clouds to provide cueing to the human AFFTRAC operator.

Table 3: *Building Descriptions and Locations*

	Description	**Latitude**	**Longitude**
Simple 1	Laundromat	32 22.736	-106 29.339
Simple 2	Gas Station	32 23.071	-106 29.269
Intermediate 1	Clinic	32 22.736	-106 29.339
Intermediate 2	Liason Office	32 22.609	-106 28.812
Complex 1	Dormitories	32 22.986	-106 28.886
Complex 2	Aquatic Center	32 22.952	-106 29.138

Each of the buildings had a center point, and two to three different sets of paths with three to five way-points associated with each path. Each way-point could be used as a static point in isolation or the entire set could be used. The full list of points is in *Appendix C*. However, the vast majority of flight test was conducted with only three ground environments (Simple 1, Intermediate 1, and Intermediate 2) and

a small sub-set of way-points due to flight time constraints. Some additional limited testing was done with other buildings in the C-12J on a limited basis as detailed in Section 4.3.3.2. However, this is expository in nature and not directly captured by metrics outlined in Section 3.7.

4.2.1.1 Simple 1.

Simple 1 is a laundromat on the WSMR. It was selected due to obvious features and simple box shape. It produces occlusion angles that are sharp and contiguous.

Figure 60: *Simple 1 Viewed from the Ground*

Figure 61: *Simple 1 Viewed from MQ-9 Sensor*

4.2.1.2 Intermediate 1.

Intermediate 1 is the clinic on the WSMR. It was selected because it has a slightly more complicated shape than Simple 1 and allowed a greater range of angles. Additionally, it has more vertical development and over-hangs that could be utilized.

Figure 62: *Intermediate 1 Viewed from the Ground*

Figure 63: *Intermediate 1 Viewed from MQ-9 Sensor*

4.2.1.3 Intermediate 2.

Intermediate 2 is the test liaison office on the WSMR. It was selected for similar reasons as Intermediate 1. It has 90 degree corners inwards towards the structure and is two-story throughout the entire building. Additionally, it has some trees scattered around.

Figure 64: *Intermediate 2 Viewed from the Ground*

Figure 65: *Intermediate 2 Viewed from MQ-9 Sensor*

4.2.2 Sortie Profiles.

Initially, 10 sorties were planned at 2 hours per sortie. Due to weather and maintenance constraints, only 4 sorties were flown for a total of 8.6 hours. Further reducing this data-set, only the MQ-9 data were pertinent to the defined metrics as the C-12J did not have a functioning Sniper ATP to record imagery and was relegated to only capturing position data. *Table 4* shows the runs within the various sorties that were executed. This is a sub-set of the planned runs which can be seen in *Appendix C*.

Each sortie begins with the virtual aircraft 6 km away from the target due east with the aircraft facing due north. Orbits were flown at 24,000 +/- 100 feet Mean Sea Level (MSL) (approximately 20,000 feet HAT). The aircraft flew at 125 knots

Table 4: *Executed Runs*

Aircraft	Run	Environment	Maneuver	Tgt Profile	Sortie	Date
C-12	6	Simple 1	AFFTRAC	Stationary	A1	17-Sep
	7	Simple 1	AFFTRAC	Moving	A2	17-Sep
	13	Intermediate 1	AFFTRAC	SOI	A2	17-Sep
	18	Complex 1	AFFTRAC	SOI	A3	17-Sep
		Complex 1	AFFTRAC	SOI 2		17-Sep
		Intermediate 2	AFFTRAC	SOI		17-Sep
		Simple 2	AFFTRAC	SOI		17-Sep
		Complex 2	AFFTRAC	SOI		17-Sep
MQ-9	19	Simple 1	4.5km Orbit	Stationary	B1	15-Sep
	20	Simple 1	6.0km Orbit	Moving	B1	14-Sep
	22	Simple 1	7.5km Orbit	Stationary	B1	15-Sep
	23	Simple 1	Cloverleaf	Stationary	B1	14-Sep
	24	Simple 1	AFFTRAC	Stationary	B1	15-Sep
	25	Simple 1	AFFTRAC	Moving	B1	15-Sep
	27	Simple 1	MQ-9 IP	Stationary	B1	15-Sep
	28	Simple 1	MQ-9 IP	Moving	B1	15-Sep
		Intermediate 2	4.5km Orbit	Stationary		15-Sep
		Intermediate 2	6.0km Orbit	Moving		15-Sep
		Intermediate 2	7.5km Orbit	Stationary		15-Sep
	30	Intermediate 1	4.5km Orbit	Stationary	B2	15-Sep
	31	Intermediate 1	6.0km Orbit	Moving	B2	15-Sep
	32	Intermediate 1	6.0km Orbit	SOI	B2	15-Sep
	33	Intermediate 1	7.5km Orbit	Stationary	B2	15-Sep
	37	Intermediate 1	MQ-9 IP	Stationary	B2	15-Sep
	39	Intermediate 1	MQ-9 IP	SOI	B2	15-Sep

+/- KCAS or approximately 180 KTAS.

4.3 Results by Category

In this section, the final results of the thesis are presented. As a reminder, each category contains distinct metrics and each metric may have a software or flight test component. Reference *Table 2* in Chapter 3 for the test matrix. The visual holding areas for the three simulated test scenarios can be seen in *Figure 66*. Roughly 75%, 50%, and 25% of the possible 360-degree area were good and can be considered "visual holding areas" in both flight test and simulation across the three scenarios. Visual areas for flight test can be seen in *Figure 67*. An orange triangle on each figure represents the approximate target location. These holding areas were generated by manual review of FMV with a target present.

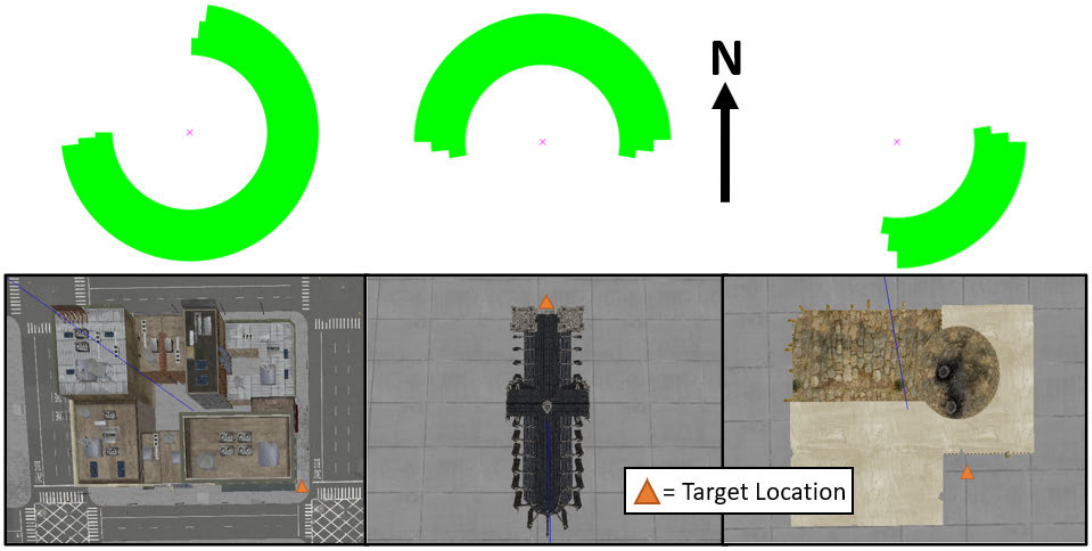

Figure 66: *Simulation - Visual Areas for the Simulated Test Environments Used*

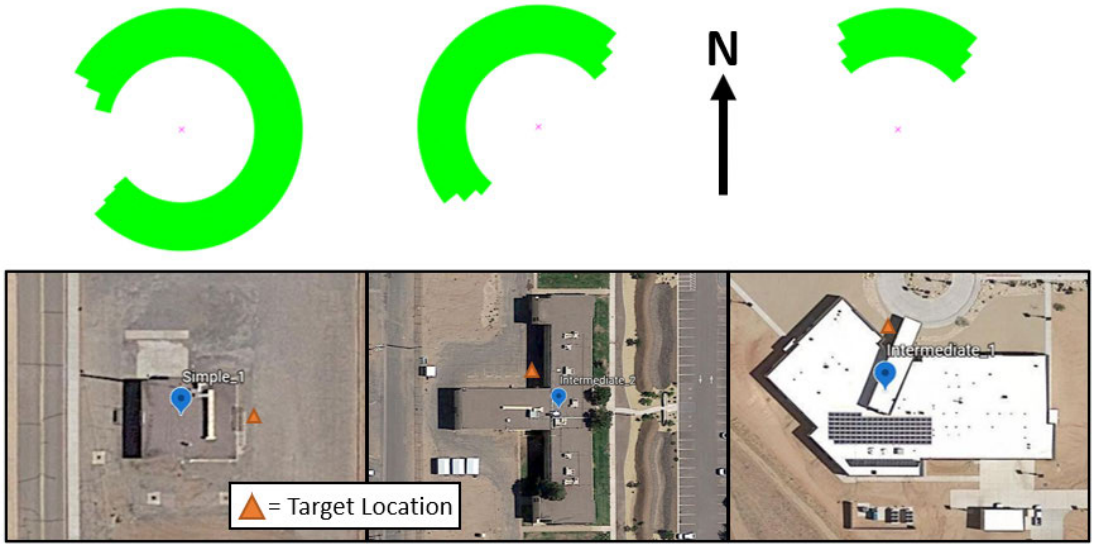

Figure 67: *Flight Test - Visual Areas for the Three Buildings Used*

4.3.1 Generation of Holding Area Densities.

There were two metrics regarding generation of holding area density. The first was a confusion matrix that shows the raw scores averaged across all holding area densities created. The second was a weighted and normalized score designed to be a single measurement of overall performance.

4.3.1.1 Holding Area Density Score.

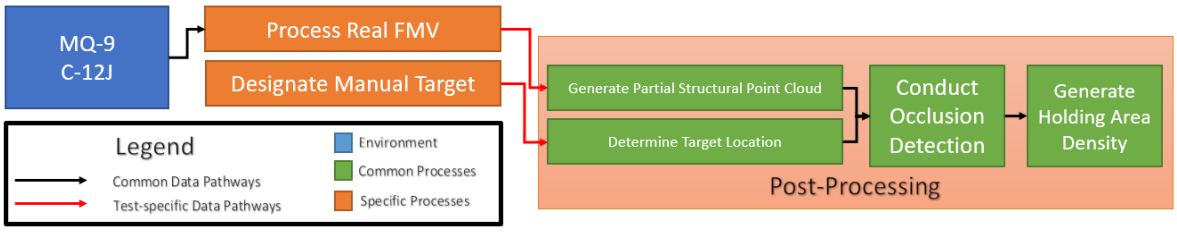

Figure 68: *Flight Test - Testing Holding Area Density Generation*

Scoring the holding area density was a functional way to evaluate the first portion of the common architecture. This flight test configuration is shown in *Figure 68*.

This evaluation was done live in simulation and in post-processing during flight test. As a reminder, what is being graded was how well the algorithm identifies occluded azimuths correctly and visual azimuths correctly as described in Section 3.7. *Figure 69* provides a visual depiction over what is being evaluated.

At each of the ground environments (whether in simulation or flight test), a target was present in the FMV feed. Three 360-degree orbits at 4.5, 6.0, and 7.5 km from the center of each target environment were flown to determine the azimuths and ranges at which the ground targets were visible and where they were occluded. Post-flight video review was performed to determine the precise azimuths, within one degree, where the target was visible to the airborne sensor. This was used to create visual holding areas which could be compared to AFFTRAC generated holding clouds as seen in *Figure 66* and *Figure 67*. These visual holding areas were constrained to be inside a ring 4.5-7.5 km from the target environment. The 6.0 km orbit was used for generation of holding area densities.

Figure 71 and *Figure 72* show that the scores are not close to 100%. While 100% accuracy is the ideal outcome, at no time does AFFTRAC have a complete structural point cloud. It only had a partial structural point cloud and by definition did miss parts of the structure that might have driven ideal holding area density generation.

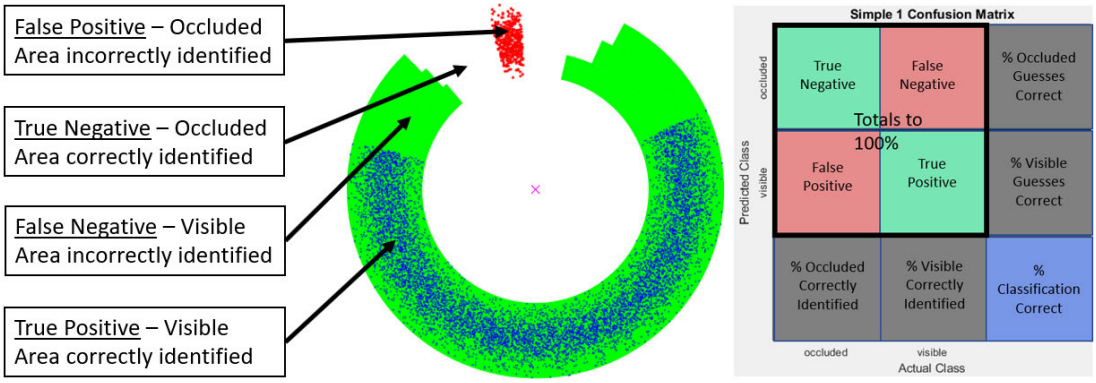

Figure 69: *Holding Area Density Confusion Matrix Visualization*

Therefore, AFFTRAC most likely would not have been able to achieve 100% accuracy with any particular holding area density. AFFTRAC attempted to make up for this by decaying holding area densities over time to hopefully produce a better average picture. Additionally, with the way the quadrant analysis works, AFFTRAC only had to do fairly well most of the time to produce correct emergent behavior.

The raw scores are in the histograms on the left side of *Figure 71* and *Figure 72*. Simulation clearly scored better on average across the three scenarios with average scores of 60.56, 60.38, and 44.26 compared to average scores of 63.99, 30.21, and 19.71 in flight test. Additionally, the simulation scores had spikes at the right part of the distribution. This suggests there were holding area densities created that are high in information and did well at rejecting occluded areas. Interestingly, they also had spikes in two of three scenarios at the low end, suggesting there were cases when information is poor. Analysis of simulation data shows that there are scenarios near the edges of the visual area where the target was barely visible where this happens. The stereo vision algorithm fails to capture significant portions of the structure. An example can be seen in *Figure 70*. Only the back-side of the tall building was populated with structural points which resulted in a high false negative rate in the holding area density. These number of poor scores in simulation relates inversely to the size of the visual area. This follows as there is a much larger area in

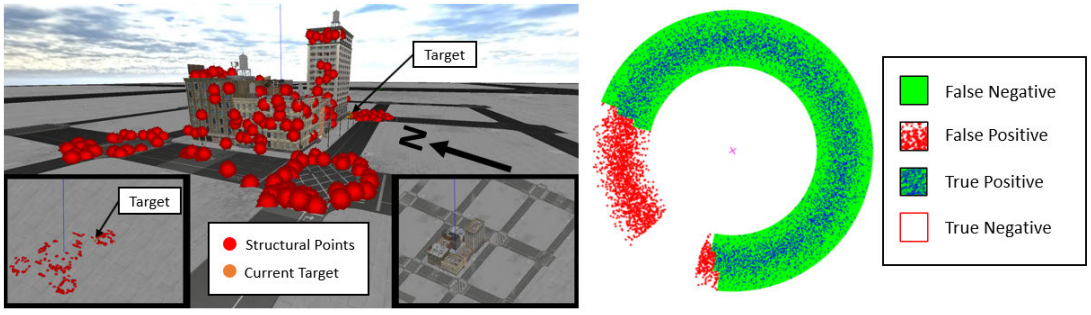

Figure 70: *Simulation - Edge of Visual Holding Area Case - City Block*

which the aircraft would have not been able to see the target and there was more room for error. Flight test tends to group around the mean suggesting the performance doesn't change much based on vantage. Further analysis of flight test results is in later paragraphs as it is not immediately obvious why the scores were as they were.

Figure 71 and *Figure 72* also show confusion matrices on the right. A legend can be found in *Figure 69*. Important numbers are summarized in *Table 5*. The percent classification correct is in green from the bottom right of the confusion matrices. It was a raw score of performance before weighing and normalization. Two other numbers pulled from the confusion matrices are in the left two boxes of the bottom row in green. These numbers were the true negative percentage and true positive percentage and correspond to how well AFFTRAC did at at correctly identifying occluded area and correctly identifying visual area respectively.

There is a clear trend where simulation beat flight test. Percent correct classification averaged 67.6% in simulation versus 56.2% in flight test. True negative averaged 39.2% in simulation versus 23.6% in flight test. True positive averaged 96.7% in simulation versus 89.2% in flight test. These numbers also validate the scoring metric. If percent correct classification was used solely as a measure of success, the gap between simulation and flight test (approx. 11 percent classification correct) would not be as large as it is in the weighted and normalized score (approx. 18 points in weighted and normalized score). However, while these numbers suggest that simulation outperformed flight test, they do little to explain why. The rest of this subsection attempts to suggest why this might be with specific examples within the common architecture ("Generate Partial Structural Point Cloud", "Determine Target Location", "Conduct Occlusion Detection", and "Generate Holding Density"). Specifically, it looks at generating the partial structural point cloud as a possible explanation.

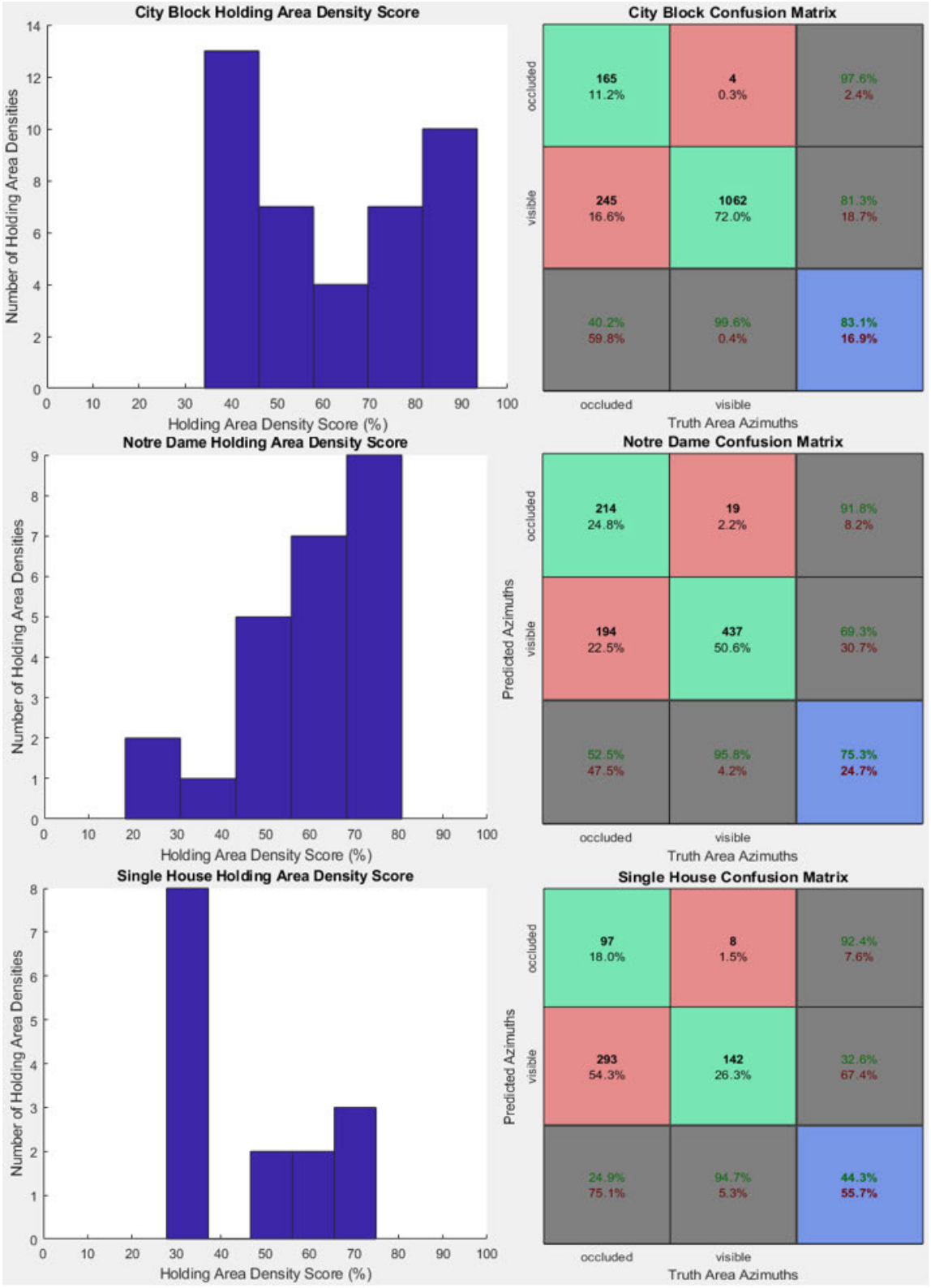

Figure 71: *Simulation - Holding Area Density Score and Confusion Matrices*

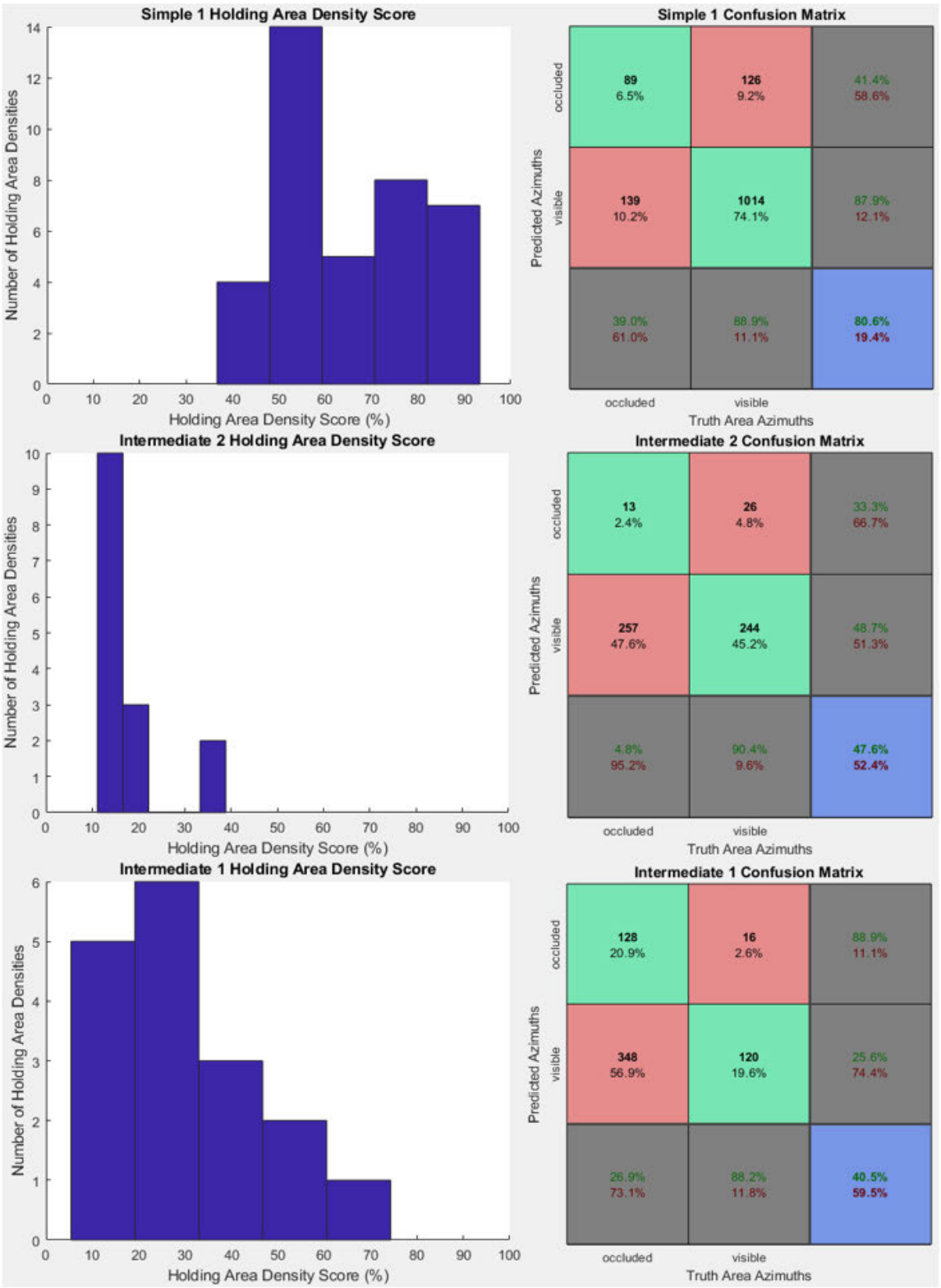

Figure 72: *Flight Test - Holding Area Density Score and Confusion Matrices*

Table 5: *Simulation and Flight Test - Holding Area Density Comparisons*

	Size of Visual Area (%)	Mean Score	Classification Correct (%)	True Negative (%)	True Positive (%)
City Block	72.8	60.56	83.1	40.2	99.6
Notre Dame	47.3	60.38	75.3	52.5	95.8
Single House	25.3	44.26	44.3	24.9	94.7
Simulation Average		**55.07**	**67.6**	**39.2**	**96.7**
Simple 1	81.9	63.99	80.6	39.0	88.9
Intermediate 2	50.1	30.21	47.6	4.8	90.4
Intermediate 1	22.3	19.71	40.5	26.9	88.2
Flight Test Average		**37.97**	**56.2**	**23.6**	**89.2**

AFFTRACs true negative rate in flight test varied widely between successive image comparisons, ranging from 0% to 100% for Simple 1 and Intermediate 1, and 0% to 29.7% for Intermediate 2. This was not the case in simulation where at least some of the occluded area was identified correctly in every holding area density created. An example of a good holding area densities created while processing Simple 1 can be seen in *Figure 74*. To understand why AFFTRAC performed poorly requires comparing the partial structural point clouds generated in simulation and in flight test.

One possibility is that the partial structural point clouds were generally more sparse when generated in flight test. This can be seen by comparing *Figure 73* with *Figure 74* and *Figure 75*. However, as the occlusion detection algorithm only required a structural point to be within 10 degrees of a cast ray, the sparsity of the points clouds was most likely not the primary culprit. In fact, if the occluded area had been 100 degrees, having only 10 points in the partial structural point could be sufficient if they were in the correct place. Also, the sparsity would not account for 0% of the occlusion area being identified correctly.

Figure 73 compared to *Figure 74* and *Figure 75* suggest the more likely culprit

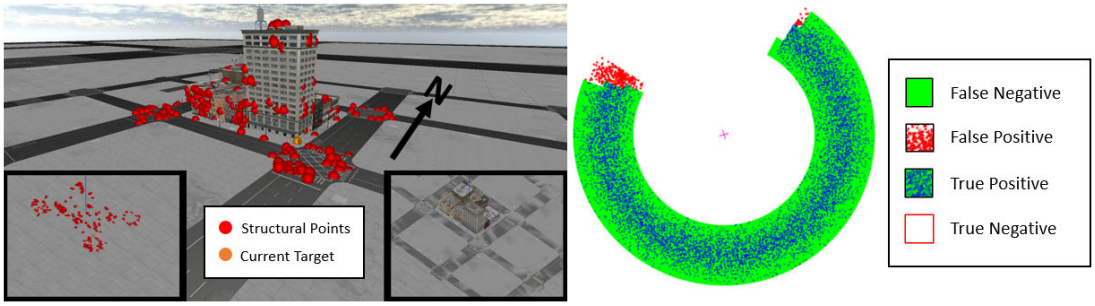

Figure 73: *Simulation - Partial Structural Point Cloud - City Block*

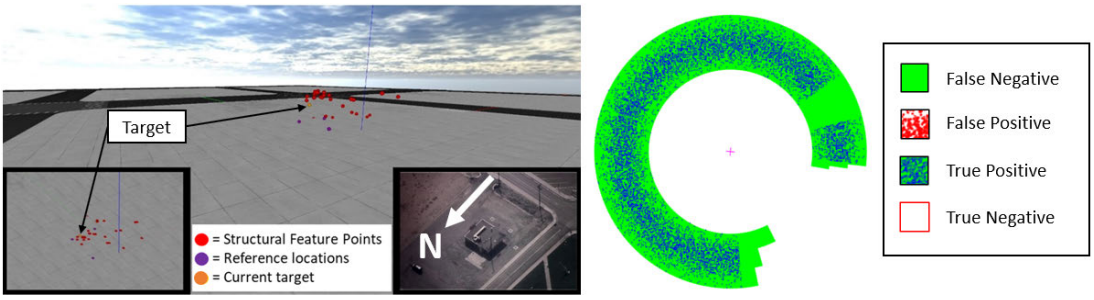

Figure 74: *Flight Test - Partial Structural Point Cloud- Simple 1*

is the rotation of the partial structural point cloud. In fact, this accounts for the occlusion area being incorrectly identified in the majority of cases. Looking at *Figure 75*, this is one such case. The video displayed in the lower right of *Figure 75* shows that the area to the NW was visible. However, because the ground plane was rotated with the ground in the air to the NW, the exact opposite happened. All of the occluded area was marked visible and a good deal of the visual area is marked occluded.

In order to try to diagnose why this rotation was occurring that did not exist in simulation, the simulation was altered to introduce position noise. The aircraft position error was added in. This error was 100 meters away from the target in one image of the image pair and 100 meters closed in the other image. Additionally, the rotation pose of the camera was not passed to the stereo vision algorithm and the algorithm was asked to estimate the camera pose rotation as is done in flight test.

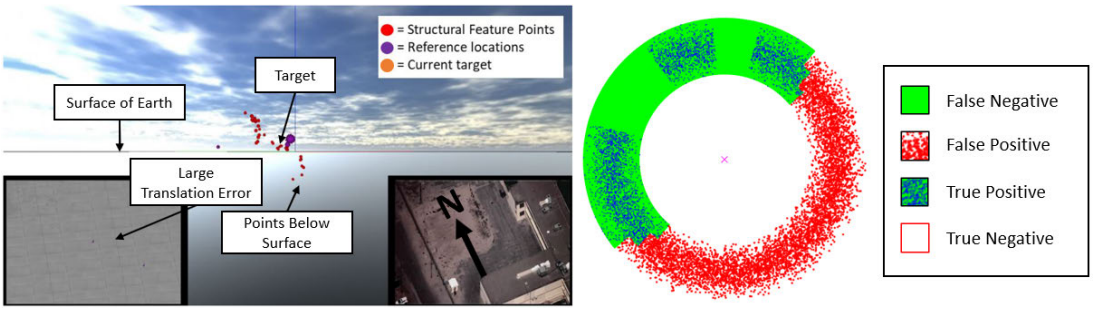

Figure 75: *Flight Test - Partial Structural Point Cloud - Intermediate 2*

The end result can be seen in *Figure 76* and *Figure 77*. This produced both an error in rotation and in translation. The point cloud density did not seem to appreciably change though. *Figure 76* shows the error in rotation. This error shows the ground canted up approximately 10 degrees. Additionally, there is an error in translation that can be seen in *Figure 77*. This error is less severe because the target is translated an equal amount in relation to the partial structural point cloud. Translation should actually make no difference assuming the rotation is correct. This suggests that this could be a possible reason for the rotation seen in flight test. This is further explored in the future work section of Chapter 5. However, it is interesting that the holding area density creation seems resistant to some error in rotation and translation. This is most likely because only inclinations of 30-60 degrees are analyzed so the errors would have to be severe as they are occasionally in flight test.

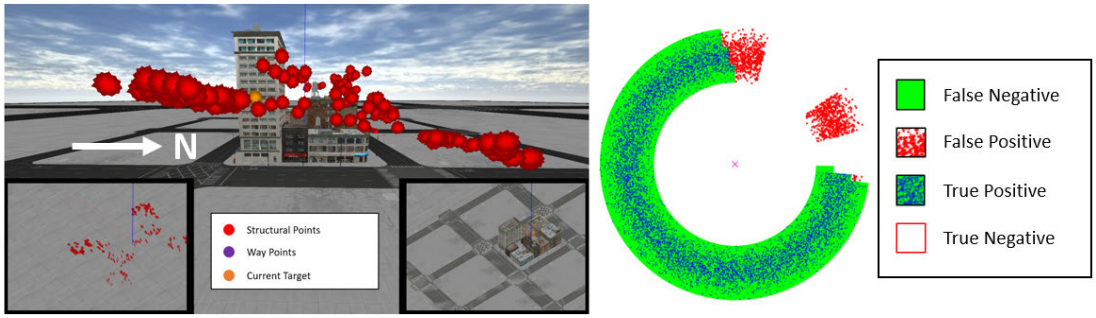

Figure 76: *Simulation - Induced Position Error Shows Rotation Error - City Block*

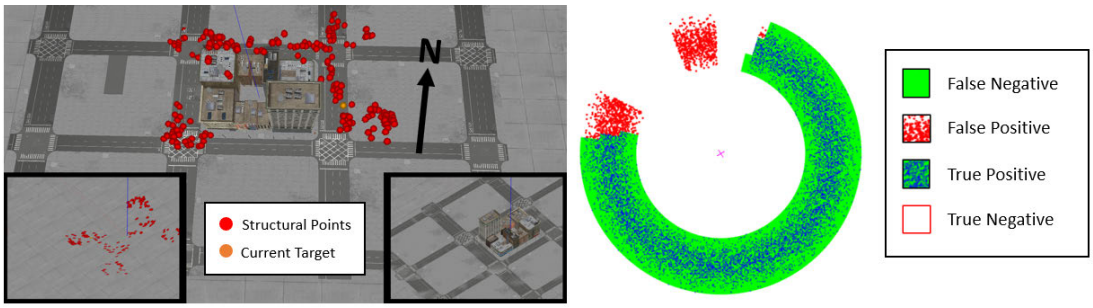

Figure 77: *Simulation - Induced Position Error Shows Translation Error - City Block*

In conclusion, simulation consistently outperformed flight test. The reason for this is that partial structural point clouds produced during flight test frequently had significant errors in rotation. Some initial testing in simulation by introducing intentional position error and not providing camera pose rotation replicated this phenomenon, albeit to a lesser degree. More development and analysis is required to fix this problem in rotation. Once fixed, existing flight test data could be re-processed to see if flight test results approach performance in simulation. This is further outlined in the Future Work section of Chapter 5.

4.3.1.2 Ability to Run in Real Time.

Each of the values in the *Table 6* is the result of averaging the algorithm execution 100 times for each test environment. The results show there isn't a huge difference in algorithm execution time between test environments. There is a direct relationship between re-projection time and outlier rejection time. This is a function of image feature quantity. The more features identified leads to more points that can be re-projected. The more points, the longer it takes to re-project and trim outliers. There is some variation beyond that, but nothing is extreme. This means the algorithm is capable of running several times a second if desired. The average total execution time is well below the 10-15 second execution interval and imposed cutoff of 200 ms.

Table 6: *Simulation Results - Average Execution Time of Main Algorithm Components*

	Partial Structural Point Cloud and Target Generation (ms)	**Outlier Rejection (ms)**	**Occlusion Detection Holding Area Density Generation (ms)**
City Block	113.1	15.1	52.3
Notre Dame	93.1	4.1	54.2
Single House	61.3	2.3	49.1
Average	**89.2**	**7.2**	**51.9**

4.3.2 Generation of Turn Commands.

This section deals solely with the flight commands displayed to the pilot in flight test and evaluating if they were correct.

4.3.2.1 Correct Turn Command.

Turns are considered correct if they would:

1. Maintain the aircraft inside a holding area density.
2. If outside, turn to return to the holding area density more directly than the other two options irrespective of any control delay.
3. Seek to orbit the target at the mean distance of 6.0 km within a standard deviation of 0.75 km.

4.3.2.1.1 Flight Test Results. For this test an MQ-9 was flown at constant altitude along a pre-defined ground track in the shape of a cloverleaf while AFFTRAC ran in the background. The ground track involved flying in and through a 360-degree holding area density, as depicted in *Figure 78*. A single run was accomplished, consisting of one complete cloverleaf maneuver. Data were collected every 5 seconds and analyzed post-flight.

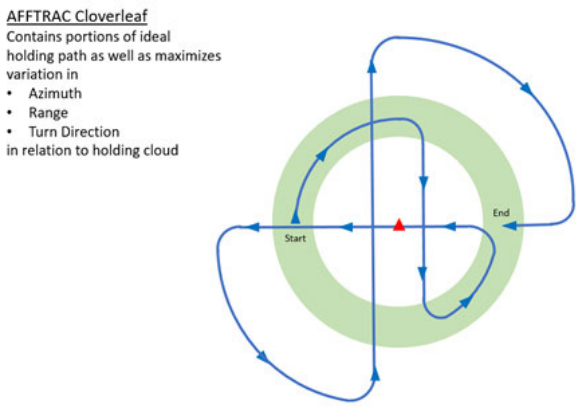

Figure 78: *Flight Test - Planned Cloverleaf Flight Path*

The ground track was programmed into the GCS, and the entire maneuver was flown using the autopilot in the MQ-9. The full cloverleaf took approximately 12.5 minutes to complete. The cloverleaf as flown can be seen in *Figure 78*. Winds were 6 knots from heading 210 and did not impact the evaluation. MQ-9 position parameters and AFFTRAC commands were recorded for post-flight analysis.

The aircraft commands were good with the exception of the red circled point within *Figure 79* in the upper left diagram. This figure shows the post-processed results of the single run performed. AFFTRAC issued 160 aircraft commands, of which 10 were incorrect. This means that overall 93.75% were correct using the grading criteria above. It is worth noting that these 10 commands would have resulted in poor orbit behavior shortly in the future. The problematic commands were issued when the aircraft was oriented perpendicular to the holding area density, as shown in *Figure 79*. These commands would have resulted in the aircraft flying out of a valid holding area, requiring a much larger correction than if a turn command were issued sooner.

Figure 79 also depicts a close-up of each problematic area. Better commands in those locations would have been an earlier turn, resulting in a merge with the radial

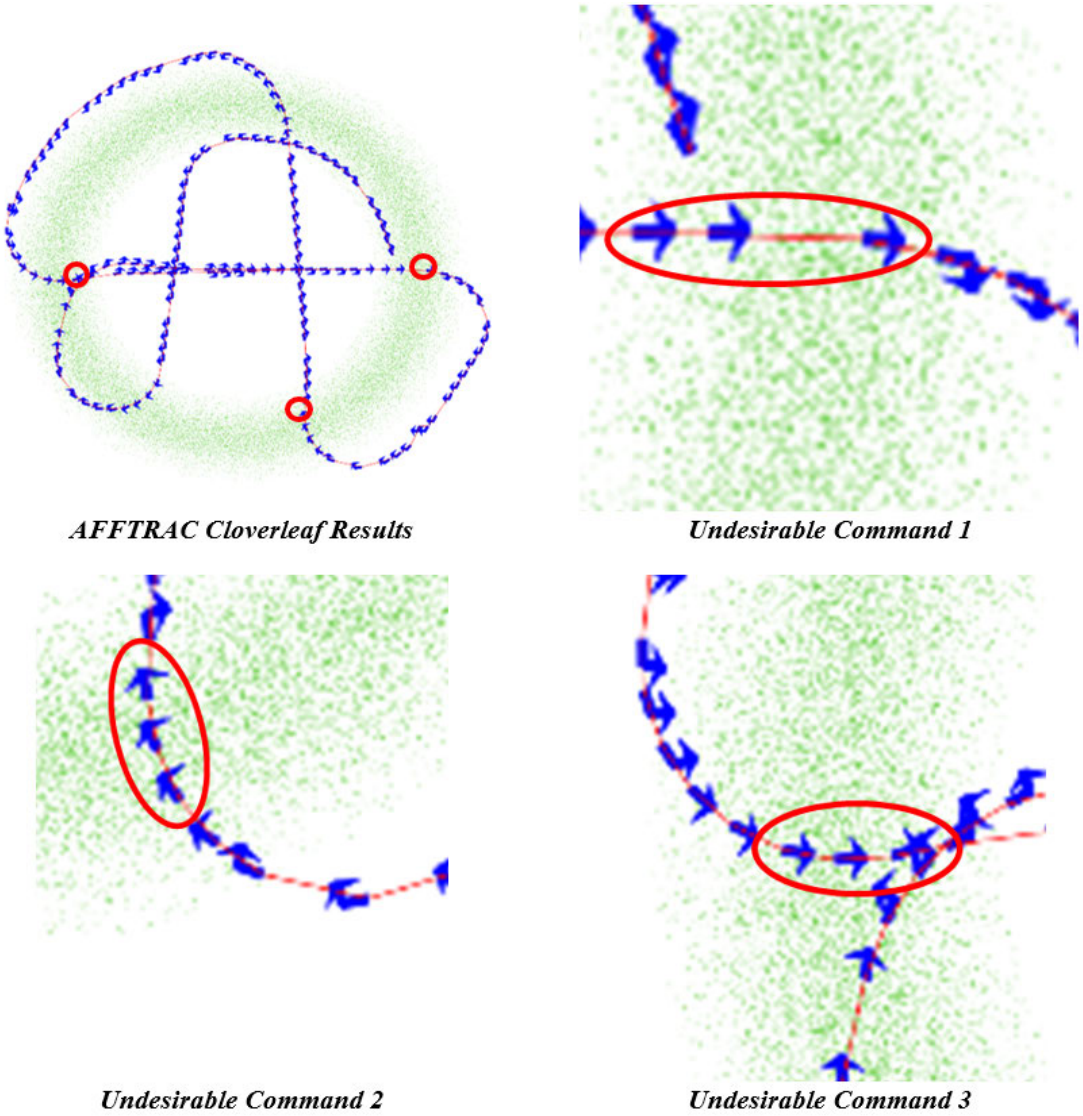

Figure 79: *Flight Test - Actual Flight Path Flown*

mean of the holding area density, rather than passing through the holding area. The quadrant analysis AFFTRAC used to generate turn commands did not consider the effects of a commanded turn. A turn right or left at any of these three error areas would have resulted in the aircraft remaining in, or at least closer to the radial mean holding area density. The inability of the quadrant analysis algorithm to turn towards the radial mean of the holding area density results in more undesirable behavior when emergent flight paths are analyzed in later sections. Specifically, it reduces the time that the aircraft is in the holding area density radial mean or in the holding area at all. It produces a constant overshooting tendency where the aircraft passes near perpendicular to the holding area density repeatedly.

4.3.3 Holding Performance of the Aircraft.

Evaluation of this category is different that in Section 4.3.1 in one major respect. AFFTRAC in both simulation and in flight test in Section 4.3.1 were comparing the first part of the common architecture. As stated in Section 3.7, the realities of flight test prevented the full execution in flight test. So there is a difference in the data feeding into the back part of the common architecture as seen in *Figure 80* and *Figure 81*.

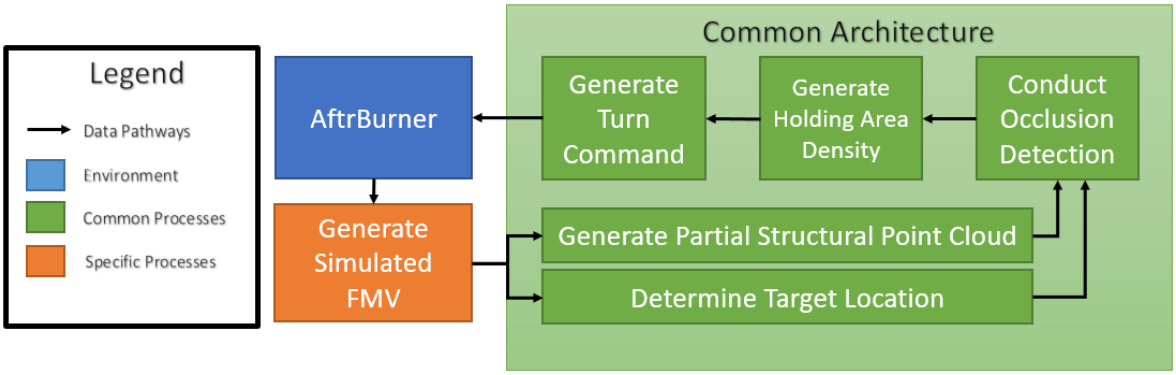

Figure 80: *Simulation - Testing Aircraft Holding Performance*

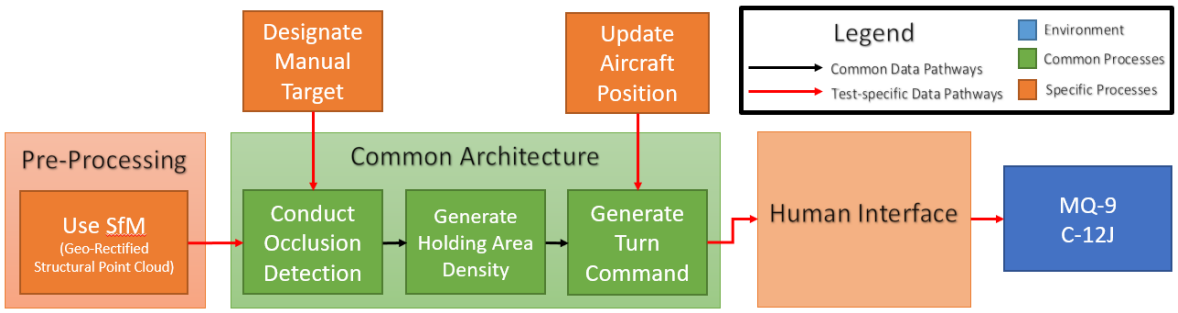

Figure 81: *Flight Test - Testing Aircraft Holding Performance*

Simulation received its data stream from the full common architecture. That is, live images were processed as they were created in real-time and turn commands were followed based on holding area densities that resulted from those images. The simulation results were a system level integration test for the entire AFFTRAC common architecture.

Flight test holding area densities were created in a different manner. SfM models were created from image pre-processed prior to a flight. This structural point clouds are a complete 360-degree geo-rectified representation of the building. They are of higher quality and contain more information than would ever be created from image pair analysis. Additionally, flight test had a human actor for comparison that simulation cannot re-create. Thus, the results in this section are separate whereas they were integrated prior.

This category has three metrics. The first is percent time in holding area. Next is percent time at optimal radius in holding area. The last is percent of the time the target is kept in visual custody (i.e. not occluded). The metrics are grouped by whether they were conducted in simulation or flight test.

4.3.3.1 Simulation.

Each of these three environments were evaluated for five runs and their performance averaged. Each run was 10 minutes of flight time. There was 30 knots of simulated wind from the west (270 at 30 knots) during all of the runs. The results are shown in *Table 7*. Additionally, examples of the aircraft paths are included in *Figure 82*, *Figure 83*, and *Figure 84*. The flight paths were similar between the runs but not identical. This is due to the fact that slight variations in the position from which the images were taken might have produced different occlusion matrices. Additionally, even if the occlusion matrices are the same, the holding area densities are sampled from probability distributions and would be slightly different in composition.

Table 7: *Simulation Results - Holding Performance of the Aircraft*

	Time in Holding Area Density (%)	**Time at Optimal Radius in Holding Area Density (%)**	**Time Target Visual (%)**
City Block	94.4	81.7	95.3
Notre Dame	91.5	74.9	97.4
Single House	75.4	52.4	96.6
Simulation Avg	**87.1**	**69.7**	**96.4**

Table 7 contains time target visual and the percentages are fairly high. All three scenarios kept the target visual at least 95% of the time. Additionally, from manual review of the runs, there were very few instances where the target was occluded for any significant period of time (greater than 10 seconds). Most of the occlusion occurred at the angular edges of the visual area. From operational experience, this sort of momentary loss of visual custody is usually acceptable. It is interesting that the city block had the worst time target visual performance. One would expect that the larger sector would reduce the time spent in turns and allow more time target visual. However, the composition of the holding area densities meant that the turn

occurred later than in either other scenario. So, even though it turned less often at the angular edges, when it did turn the time of visual custody loss was greater.

Time in holding area averaged 87.1% across all three scenarios. It is worth noting that the percentages were high partially due to the decay of the holding area density. It was highly probable that there is some number of points within 350 m in all quadrants of the aircraft even when the aircraft left the visual area azimuths. This was due to the decay described in Section 3.5.2 and the performance seen in Section 4.3.1. Nearly all of the time outside of holding area density was due to being too close or too far along the ground radius. *Table 7* shows that sector visual area size is directly related to time in holding area. The larger available sector in the city block provided more time for the aircraft to settle into an arc around the target with fewer turns in comparison to the other two environment. The same relation held when comparing Notre Dame to the single house. It is notable that the single house scenario had a markedly lower score. The performance is further degraded by the behavior identified in Section 4.3.2. This can be seen in *Figure 82*. The aircraft continued straight two separate times perpendicularly through the holding area density instead of seeking to intercept the mean. If it had done so, it would most likely have settled into a better sector hold.

Time at optimal holding radius in the holding area density followed the same trend as time in holding area density. The reasoning behind is exactly the same. It is worth reviewing why these figures are important. Ground distance corresponds directly to image size, image quality, stability of focus, and predictability of visual occlusion. When an aircraft fails to maintain a constant radial distance, it decreases the quality of the imagery produced. Failing to maintain an "optimal" holding distance as defined in Section 3.7 causes slight degradation from operational experience. Leaving the holding area density by getting too close or too far is a more serious problem. It

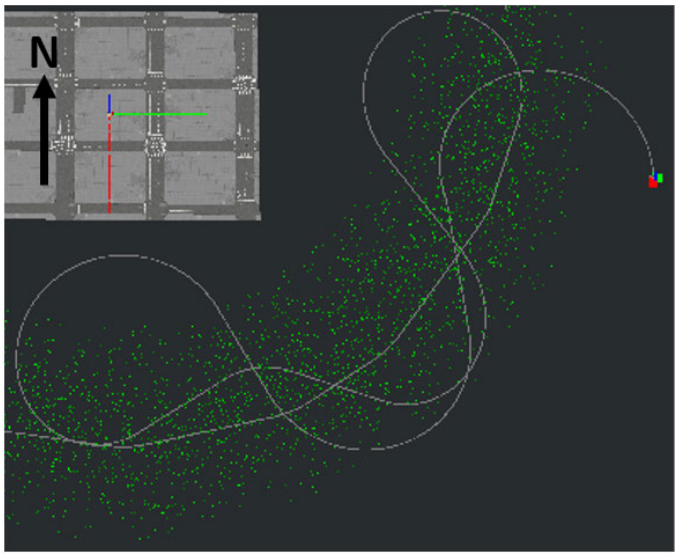

Figure 82: *Simulation - Flight Path at Approx. 4 Minutes - Single House*

can necessitate re-focusing of the camera and increase the workload of the sensor operator.

Consider the couple of examples of the flight path produced during these runs. *Figure 82*, *Figure 83*, and *Figure 84* show three such examples. *Figure 83*, *Figure 84* are from the city block and the Notre Dame respectively. The interesting thing to see in these pictures is the emergent behavior. Both of the holding paths shows that it took AFFTRAC a certain amount of sector distance to settle into the radial mean. Additionally, it is worth looking at the characteristics of the holding area densities in a state of decay. There is a noticeable difference in *Figure 84* between the density where the visual area decayed and the density of the occluded area. The threshold in the quadrant analysis between the forward and rear quadrants can be seen in action. This differential caused the aircraft to turn around back to the visual holding area even though there was density ahead of it. This happened even when the density difference was not as obvious as in *Figure 83* but the density difference was sufficiently high to trigger the correct behavior. *Figure 82* has already been

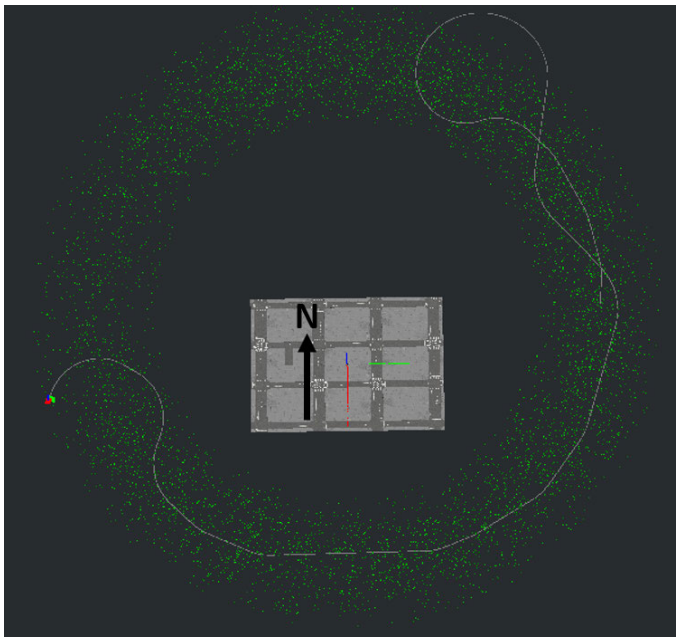

Figure 83: *Simulation - Flight Path at Approx. 6 Minutes - City Block*

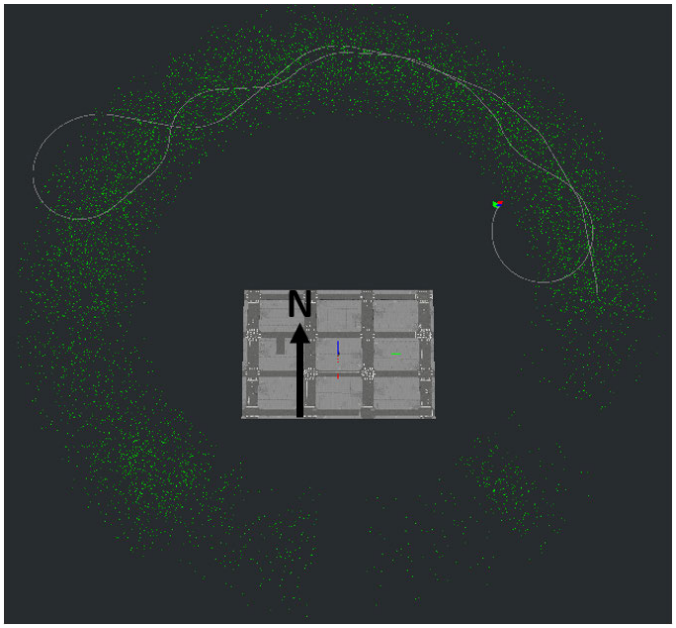

Figure 84: *Simulation - Flight Path at Approx. 5 Minutes - Notre Dame*

discussed earlier but it is worth noting again the undesirable behavior that resulted from the inability of the quadrant analysis algorithm to intercept the radial mean from a perpendicular approach. This behavior was aggravated as shown in the flight test result section next.

4.3.3.2 Flight Test Results.

There were two different sets of flight test results to consider. The first were results from the MQ-9. The MQ-9 had an EO/IR pod during the flight test so it was possible to review the video footage and determine whether or not the target was in visual custody. Additionally, it was possible to grade the profiles in comparison to a MQ-9 IP run. This allowed AFFTRAC to be directly compared to the best possible tactical executors in existence, human operators. The second are results from the C-12. It was not equipped with an EO/IR pod so video review was not possible nor was an MQ-9 IP run possible. The flight test team elected to only collect flight path information because of it. Time in holding area density, time at optimal radius, and time target visual cannot be verified. Thus, the results have been separated.

In both cases, AFFTRAC ran using pre-processed SfM models. The MQ-9 required manual position and heading updates to be typed into the AFFTRAC laptop as quickly as possible and displayed turn directions to the pilot on a GCS monitor. The C-12 received GPS updates from a GPS puck automatically. However, the turn commands had to be verbally relayed to the C-12 pilot by the AFFTRAC operator. In both cases, the airspeed was faster than in simulation (180 KTAS vs 140 KTAS), the control delay was longer (2 seconds vs 0 seconds), the time to achieve standard rate turn was longer, position uncertainty was present, and the holding area density had a radial range of 4.5 km to 7.5 km. The holding area density range should have been 4.5 km to 8.0km based on the 180 KTAS airspeed to allow one turn radius

but this was not possible due to airspace constraints. All of these considerations are important when analyzing the three metrics and emergent pathing.

4.3.3.2.1 MQ-9 Results. This test was conducted using an MQ-9 as the test aircraft, a single ground structure as the occlusion environment, and a single person as the target. Separate test runs were conducted for both a stationary and a moving target, with the aircraft pilot following a 360-degree wheel orbit around the occlusion, with an MQ-9 instructor pilot flying established techniques, tactics, and procedures (TTPs), and with the aircraft pilot following guidance from AFFTRAC. The precise locations of the occlusion and target are specified in *Table 8* and depicted in *Figure 85*. For the stationary case, the target remained at position "S1_A1" for the entire test run. For the moving case, the target moved between points "S1_A1", "S1_A2", and "S1_A3", beginning movement after being at each point for 4 minutes. All runs ran for 10 minutes.

As implemented, AFFTRAC could not communicate directly with the MQ-9 GCS to incorporate automatic position updates. Instead, position updates were keyed in manually. To accommodate manual coordinate entry the turn command update rate was set at one update every seven seconds. The added effect of manual coordinates added several seconds of latency which would not be present in an integrated system. The exact latency in the coordinate update system was not measured. Additionally, although there was no holding area density to follow in the case of the 360-degree orbit and the MQ-9 IP run, their ability to stay within a holding area density was compared against the visual area within 4.5 km and 7.5 km at the current time step.

The values in *Table 9* lack statistical rigor due to the small sample size, but anecdotal observations from the specific test performed may still be made. In a limited, controlled test scenario with an ideal structural point cloud (generated by SfM), AFFTRAC demonstrated the capability to outperform a 360-degree wheel orbit

Table 8: *Flight Test - Target Location and Path - Simple 1 Stationary*

Profile	**Point**	**Time (min)**	**Latitude**	**Longitude**
Center	**Simple_1**	n/a	32 23.054	-106 29.001
S1_A	**S1_A1**	0:00	32 23.054	-106 28.995
	S1_A2	4:00	32 23.058	-106 28.995
	S1_A3	8:00	32 23.058	-106 29.001

Figure 85: *Flight Test - Target Locations - Simple 1*

and at least match or outperform a qualified instructor pilot when trying to maintain visual custody. However, the MQ-9 IP (using established TTPs) maintained a higher quality hold in terms of staying within the holding area density and optimal distance. Additionally, the MQ-9 IP intentionally lost visual at the edges of the available sector a couple of times for the express purpose of determining exactly where occlusion occurred. The 360-degree hold was a perfect circle so its time in holding area equaled its time at optimal distance equaled its time target visual. However, its time target visual suffered because there was no attempt to maintain visual custody of the target and any time visual was due to luck. This demonstration showed the potential benefit of this system although more robust testing would be required to draw conclusions about an operationally representative implementation of AFFTRAC.

Aircrew comments relating to AFFTRAC-commanded holding, flight profile, track

Table 9: *Holding Area Density Comparisons*

	Time in Holding Area Density (%)	**Time at Optimal Radius in Holding Area Density (%)**	**Time Target Visual (%)**
Simple 1 (Stationary)			
360-Degree Hold	76.3	76.3	76.3
MQ-9 IP	84.2	45.3	92.1
AFFTRAC	42.6	24.1	100.0
Simple 1 (Moving)			
360-Degree Hold	88.8	88.8	88.8
MQ-9 IP	90.1	51.6	100.0
AFFTRAC	48.3	28.2	100.0

stability, and image quality were also collected for each sortie. The aircrew observed that AFFTRAC-commanded turn directions were correct, but the command frequency was low and correct turn commands were delayed, which resulted in poor maintenance of the desired holding area density. The turn command latency may have resulted in the observed undesirable pathing about the prescribed 4.5 7.5 km range (*Figure 86*). Comments also suggested this behavior was amplified by the command latency inherent in the MQ-9 system when operating using Beyond LOS communication (i.e. satellite link, 1.5-2.0 seconds of latency) and by the higher than typical airspeed for ISR operations: 125 KCAS (approximately 180 KTAS) as opposed to 105 KCAS (approximately 140 KTAS as in simulation). Each of these factors combined to increase the ground distance traveled by the test aircraft between command updates. Improving command frequency, reducing command time delay, and lowering airspeed would result in a decrease in the ground distance traveled between updates and would likely improve ability to maintain a specified holding area density. Current time delay

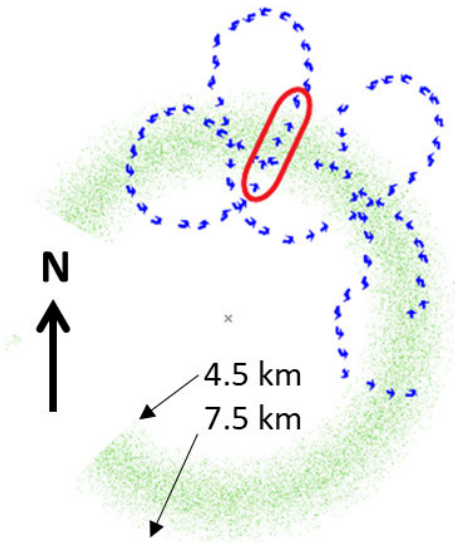

Figure 86: *Flight Test - AFFTRAC Commanded Flight Path - Simple 1*

inherent to the MQ-9 command link structure was beyond the scope of this investigation and the airspeed was within a tactically representative window; however, command frequency can be addressed in future implementations of AFFTRAC.

The aircrew also suggested potential improvements to the turn command logic. While the turn logic generally gave correct directional commands, it did not always result in optimal performance. This was noted in two areas specifically. The first resulted when the aircraft re-entered the holding area density on a heading perpendicular to the desired orbit path. In these cases, AFFTRAC issued straight-ahead command guidance which resulted in the aircraft exiting the holding area density on the opposite side before attempting to turn the aircraft. An example of this is depicted in *Figure 86* above (circled in red). Comments from aircrew suggested this could be improved by incorporating flight path prediction with respect to the holding area density instead of relying on the simple greedy search which resulted in no turns during these perpendicular encounters.

The second area of aircrew-suggested improvement to turn logic was related to the

MTS sensor geometry. On multiple occasions the sensor track could not be maintained due to the target passing through the nadir of the sensor. This behavior occurred during turns at close ground range to the target when the sensor depression angle was greater than approximately 80 degrees. AFFTRAC commanded turns were specified as standard rate only. Aircrew suggested that if AFFTRAC considered the MTS depression angle and modulated its turn rate or direction to avoid placing the target at the sensors nadir this behavior could be mitigated. AFFTRACs guidance should consider relative depression angle and the full envelope of aircraft maneuverability to avoid nadir and masking.

Overall, the aircrew-recommended improvements to AFFTRAC centered on improving its ability to maintain the specified holding area density and prevent sensor problems such as masking and nadir. Suggestions such as predictive flight pathing, increased command frequency, and consideration of sensor depression angle were directed towards addressing discrepancies observed in AFFTRACs ability to perform these tasks.

4.3.3.2.2 C-12J Results. This test was conducted using the C-12J as the test aircraft. It was not equipped with an EO/IR pod so video review was not possible nor was an MQ-9 IP run possible. This means that the only data collected was flight path information. Time in holding area density, time at optimal radius, and time target visual cannot be verified. Separate test runs were conducted for both stationary and moving targets, and with the aircraft pilot following guidance from AFFTRAC. The precise locations of the occlusion and target are not specified in this thesis in detail beyond what is outline in *Appendix C*. The test runs are included to show more AFFTRAC pathing behavior beyond what was accomplished in the MQ-9.

The C-12 received GPS updates from a GPS puck automatically. The turn commands had to be verbally relayed to the C-12 pilot by the AFFTRAC operator. This

introduced a different type of control lag than the MQ-9. However, the average response time from AFFTRAC command issued, to command actuated, to full standard rate turn did not seem appreciably different, although it was not directly measured.

Looking at *Figure 87* the problems inherent with the quadrant analysis become even more apparent. Every one of the paths had multiple instances where the aircraft ended up on a trajectory perpendicular to an holding area density median arc and flew through it without turning. This created an oscillatory behavior that sometimes dampened, sometimes maintained the same amplitude, and sometimes increased in amplitude. Other odd behaviors were noticed such as tight holding in Complex 1 - SOI 2, Simple 1 - Stationary, and Intermediate 2 - SOI *Figure 87*.

Before finishing Chapter 4, one last comparison is worth making. The effect of a larger control time delay, lower control frequency, faster airspeed, and a holding area density that did not have sufficient room for a 180-degree standard rate turn within the 4.5 - 7.5 km radial range can be seen when comparing flight paths from simulation in Section 4.3.3.1 (*Figure 82*, *Figure 83*, and *Figure 84*) to the flight paths (*Figure 87*). The flight path in simulation "settled" into the radial mean of the holding area density given a large enough sector whereas the flight test flight path did not. The oscillatory nature of the flight path is a direct consequence of these differences.

The previous comments from the MQ-9 crew in section 4.3.3.2.1 still hold true as solutions to the behavior seen in the emergent pathing. And, if AFFTRAC is to tackle more complicated scenarios in future development, these changes will undoubtedly need to be implemented in some capacity. However, the performance in simulation may be adequate for simpler scenarios if AFFTRAC is directly interfaced into the GCS and flown at a slower (and more ISR representative) airspeed. Flying closer to 140 KTAS, allowing higher command frequency, and reducing command delay might allow an MQ-9 to achieve flight paths more in line with simulation. Another

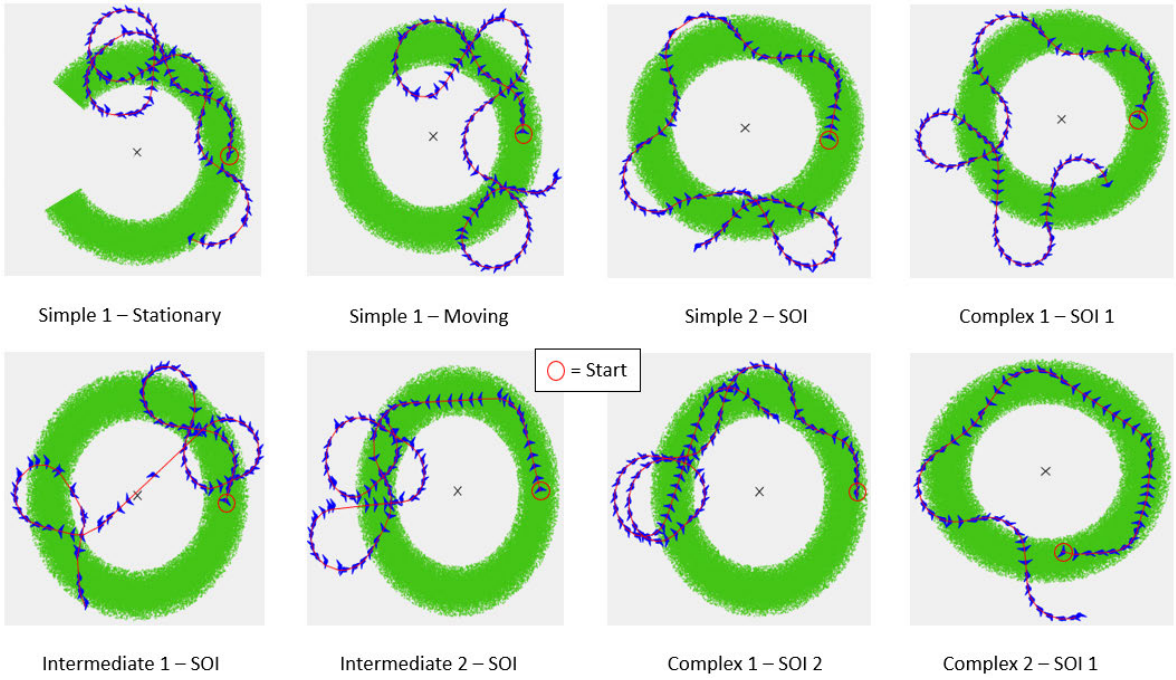

Figure 87: *Flight Test - Flight Path for Various Scenarios*

possibility is that increasing the radial range of the holding are while flying 180 KTAS (to match an actual turn radius) might have dampened the oscillatory nature of the emergent flight path.

Overall, the holding would have most likely allowed good visual custody, but the time in holding area density and time at optimal radius would have suffered greatly due to oscillatory pathing. This oscillatory behavior might be improved by direct integration into the GCS with a slower airspeed. However, this would not fix the initial problematic turn commands as seen in Section 4.3.2.1.1 and would only lessen the severity of the oscillations in future time-steps. Incorporating some sort of future path reward function is necessary and possible given existing data structures. This will be expounded upon in Chapter 5.

V. Conclusions and Future Work

5.1 Conclusions

The research objectives as described in Chapter 1 of this thesis are defined in terms of resultant behavior from an automated control system. They were designed so that the system's emergent behavior can be judged in both a simulated and real-world test flight comparatively. With this in mind, the system should be able to:

- Fly an initial 360-degree orbit around a initial set of coordinates.
- While maintaining an orbit, build an internal three dimensional representation of the ground environment from the full motion video and telemetry taken in.
- Identify a particular object of interest and determine a hold area that allows good distance, look angle, and prevents visual occlusion.
- Execute appropriate aircraft holding to maintain that area.
- Above all, maintain visual custody of the target.

This was evaluated with these categories of evaluation metrics;

1. **Generation of Holding Area Densities**
   (a) Holding Area Density Score
   (b) Ability to Run in Real Time
2. **Generation of Turn Commands**
   (a) Correct Turn Command
3. **Holding Performance of the Aircraft**

(a) Percent Time in Holding Area

(b) Percent Time at Optimal Radius in Holding Area

(c) Percent Time Target Visual

Overall, the system was able to meet the intent of the research objectives. The system demonstrated capability to fly an initial 360-degree orbit around an initial set of coordinates. It did this by building a 360 degree holding area density when no target was present. While flying to create sufficient parallax during this orbit, it took sequential images and created a internal three dimensional representation of the ground environment in the form of a partial structural point cloud. The next research objective was to create a holding area that allowed good distance, look angle, and prevents visual occlusion.

In simulation, the partial structural point cloud drove creation of holding area densities that identified the occluded area correctly 39.2% of the time and the visual area correctly 96.7%. They correctly characterized the total possible holding area 67.6% of the time and generated a mean score of 55.1%. The scores reflect the assumption that false negatives were worse than false positives. The run time for this common architecture was below the 200 ms cutoff with an average of approximately 150 ms. Flight test performance was worse than simulation in creating holding area densities. The holding area densities identified the occluded area correctly 23.6% of the time and the visual area correctly 89.2%. They correctly characterized the total possible holding area 56.2% of the time and generated a mean score of 37.97%. The suspected reason for this discrepancy is that the partial structural point clouds used to create the holding area densities suffered frequently from significant errors in rotation.

The next research objective of executing appropriate aircraft holding to maintain that holding area was partially accomplished. A cloverleaf pattern was flown ignoring

the commands AFFTRAC supplied to judge the effectiveness of command produced during quadrant analysis. 93.75% of the commands produced correct behavior. However, the remainder of the commands produced undesirable future behavior. Any time the aircraft was approaching the holding area density roughly perpendicular the the median radial arc of the density, it continued going straight instead of turning to intercept the radial mean of the density. This later produced emergent behavior where the aircraft path oscillated around the mean instead of flying down it as desired.

In simulation, quadrant analysis exploitation of holding area densities created from partial structural point clouds allowed the aircraft to stay within the holding area density 87.1% of the time on average. The aircraft was at an optimal radius of 6.0 km with a SD of 0.75 km 69.1% of the time on average. In flight test, the holding area densities were created from a pre-processed SfM complete structural point cloud. This was compared against a 360-degree orbit and an MQ-9 IP run against the same target. In the MQ-9, this allowed the aircraft to maintain the holding density an average of 45% of the time and within optimal an average of 26% of the time. The 360-degree hold allowed the aircraft to attain an average of 82% of the time for both. The MQ-9 IP maintained the holding density an average of 87% of the time and within optimal an average of 48% of the time. Here, AFFTRAC was clearly worse in comparison to either the 360-degree orbit or the MQ-9 IP.

This ties into the last objective, which was to above all maintain visual custody of the target. In simulation, AFFTRAC maintained visual of the target 96.4% of the time. In flight test, AFFTRAC maintained visual custody of the target 100% of the time compared to an average of 82% for the 360-degree orbit and 96% of the time for an MQ-9 IP. This means that AFFTRAC tied or surpassed both the 360-degree and the MQ-9 IP in visual custody. However, the data-set was severely limited in flight

test and the scenarios were simplistic. But, given that a primary motivation was to allow automation in simplistic scenarios, which make up the bulk of flight time, this result is promising. Additionally, it is interesting that AFFTRAC was able to achieve a good visual custody even with relative poor ability to stay within the holding area density. However, aircrew comments reflect that the actual tactical performance was lacking as it would have caused a high work-load for the sensor operator. The cause of this was irregular holding distances and oscillatory aircraft path.

Overall, the AFFTRAC common architecture shows promise as a framework for future development. The concept of a holding area density is a novel adaptation of a value map for automatic ISR holding. However, the current common architecture used to create it and exploit it needs improvement. The error in rotation in partial structural point clouds is a fundamental problem that needs to be solved. If the rotation problem is solved, then real-world use will see performance more on par with simulation, which was adequate with the partial structural point clouds made from simulated FMV. The quadrant analysis algorithm used to produce holding commands and emergent holding paths was good enough to maintain visual custody. However, the times that the commands were wrong produced oscillatory flight paths. AFFTRAC was better than a both a 360-degree orbit and an MQ-9 IP at maintaining visual custody but was worse at staying in the holding area density. Even with this trade-off, AFFTRAC was superior overall to the 360-degree orbit. However, with the MQ-9 IP, aircrew comments accurately reflect that overall AFFTRAC performance is lacking due to high sensor operator workload. The next section outlines future work to improve the performance of AFFTRAC or a similar system in the future.

5.2 Future Work

This final section outline suggestions for future work on AFFTRAC. Sections 5.2.1 - 5.2.5 suggest ways to improve the existing common architecture by either fixing components or improving the existing algorithms. Sections 5.2.6-5.2.8 suggests areas of research in which parts of the common architecture might be entirely replaced while still maintaining the basic data structures contained within.

5.2.1 Solve Rotation Errors in Partial Structural Point Clouds during Flight Test.

This is the main problem that needs to be solved prior to continuing work on AFFTRAC. Several possibilities exist:

- The current stereo vision algorithm is not receiving camera pose rotation information from the aircraft. Integrating this rotation matrix may be sufficient to fix the rotation problem inherent in the partial structural point clouds.

- The telemetry information used in flight test may not be accurate. The position information might need to be passed through a Kalman filter to better localize the exact position.

- There may be another unknown error inherent in the OpenCV libraries used for stereo vision. The extreme distance between the camera locations, extreme distance to the object being imaged, and small field of view may not be suitable to the OpenCV algorithms. They were authored with much smaller (i.e. personal) distances in mind.

- The airborne imagining system may need to be properly calibrated. While the virtual camera was calibrated in simulation, the actual sensor was not. This

was due to the long focal length. A checkerboard could not be held in front of the aircraft sensor on the ground because it was not the same lens and detector used at altitude. It may be necessary to paint a checkerboard on a billboard in a field and capture footage of it at representative altitudes and temperatures to properly calibrate the camera at the beginning of every sortie. This may account for errors in re-projection as well.

5.2.2 Integrate Target Identification and Localization.

Target identification from FMV is currently not implemented in AFFTRAC. A Convolutional Neural Net (CNN) could most likely solve this problem. Simply drawing a box around the correct target is well within current CNN's capabilities. AFFTRAC can localize a target in relation to a partial structural point cloud if pixel space is marked on two sequential images. This would allow AFFTRAC to determine target location purely from two dimensional images with no manual injection.

5.2.3 Explore Disparate Holding Area Densities.

AFFTRAC is currently capable of producing disparate visual holding areas. No testing was done to determine performance with disparate visual holding areas. The currently implementation would have either "bridged" disparate areas and treated them as a contiguous holding area or simply remained in one of the holding areas. AFFTRAC could be modified to select a holding area that is larger to improve time in holding area density and time at optimal range metrics as those are directly tied to the size of the holding area. The quality of the flight path is also a function of holding area density arc size.

5.2.4 Alter or Improve Quadrant Analysis Algorithm.

The problems with the current quadrant analysis algorithm have been covered several times in this thesis. Possibilities for improvement include:

- Allow the quadrant analysis to produce graduated outputs other than simply straight, standard rate left, or standard rate right.
- Fix the existing quadrant analysis algorithm so that it is capable of recognizing when it is perpendicular to a mean radial arc and prevent it from continuing straight.
- Improve the quadrant analysis algorithm by looking several time steps into the future and aggregate the total future discounted reward before making a turn decision.

5.2.5 Account for Depression Angle.

AFFTRAC-generated holding clouds did not account for depression angle, which meant that they did not take into account changing occlusion of the target based on range. This phenomenon is illustrated in the visual holding areas, where there are variations in the azimuths a target is view-able between ranges of 4.5-7.5 km. Instead, AFFTRAC considered an azimuth entirely occluded if a ray at any depression angle at that azimuth intersected a point in the structural point cloud. Because of this, AFFTRAC often marked a amount of good holding area as unusable. If AFFTRAC considered depression angle (i.e. range), a larger portion of the visual holding area would be made available for MALE-ISR operations. More complex scenarios will require this capability. For instance, a person underneath an overhang.

5.2.6 Use Machine Learning to Create Holding Area Densities Directly.

Holding area densities are currently being created by creating partial structural point clouds and exploiting them in relation to a target. It is feasible that Machine Learning could use a CNN embedded in a more complex neural net to produce the holding area density directly. The downside is that this approach would require training on an extensive data-set to generalize. The data-set would have to be marked appropriately so that every image was tied to a visual holding area. The geometric algorithms currently used by AFFTRAC require no prior exposure to the building to build an internal representation of the target environment.

5.2.7 Simultaneous Localization And Mapping (SLAM) to Create Structural Point Clouds.

A related area of research to create three dimensional representations of the environment from two dimensional images is monocular SLAM. The main difference is that in SLAM the pose of the camera is not well localized. While the camera pose can be passed to the stereo vision algorithm and refined with other sensors (odometry, accelerometer readings, and signal triangulation [20][70][40][43][52]) AFFTRAC demonstrates that position and pose estimation errors can results in errors in rotation and scale in point re-projection. Monocular SLAM [55][19][40][20] [70] is useful because it localized and geo-rectifies in a manner similar to SfM but does it on-line

An example of a point cloud generated by SLAM can be seen in *Figure 88*. SLAM techniques have been refined to the point that they can run real-time on an average cell phone processor with 1 cm resolution in indoor environments [29]. Additionally, modern techniques are capable of generating large and varied point clouds [29][69][66]. They achieve fairly reliable loop closure even with large and varied paths [29][69].

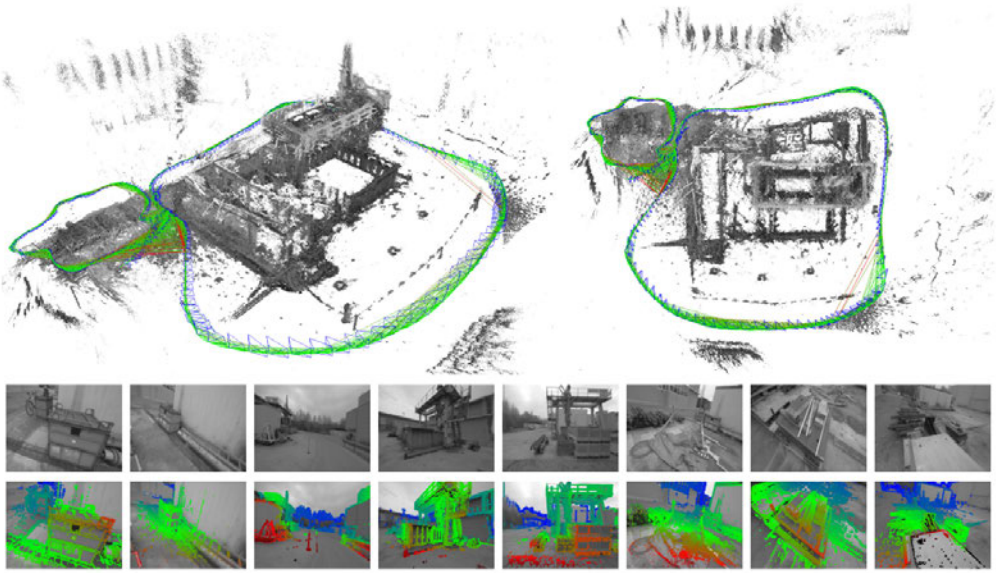

Figure 88: *LSD-SLAM: Large-Scale Direct Monocular SLAM [29]*

There are methods that also include real-time plane detecting within the generated point clouds [69][66]. Unfortunately, most of the research is done either in internal environments [69][66], foot-traffic exploration [29][69][66], or very low altitude [20]. Thus, the problem is to make SLAM work on a long-focal length aerial aperture moving at hundreds of feet a second. Additionally, the point clouds produced by SLAM would have to be heavily trimmed to keep the point count small (1,000) so that AFFTRAC could continue to run in real-time.

5.3.8 Deep Reinforcement Learning (DRL) to Exploit Holding Area Densities.

Another way to exploit a value map is through DRL [56]. AFFTRAC generated each initial reward through ray tracing and created a static value map. Deep reinforcement learning could directly exploit that value map and allow an agent to learn how to navigate it. The utility of each action would be evaluated through successive

iterations. Each iteration would initialize the aircraft to a random place and allow it to make random decisions. The sum total of decisions would aggregate into a cumulative reward function (similar to *Equation 14*) that would effectively "grade" the performance of the agent. By iterating a sufficient number of times, decisions at each step would be refined to favor the correct action to result in a maximal end value [56]. The metrics used in this thesis could serve as a starting point to grade the agent's behavior.

Bibliography

[1] Federal Air Administration. *Airplane Flying Handbook (FAA-H-8083-3B)*. 2020.

[2] Heist Andromanan. *Regard 3D*. 2017. URL: https://www.regard3d.org/.

[3] Song Ho Anh. *2D Image Processing - Ho.pdf*. 2006. URL: http://www.songho.ca/dsp/convolution/convolution2d{_}example.html (visited on 10/03/2019).

[4] Lars Arge et al. "Cleaning massive sonar point clouds". In: *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems* 2 (2010), pp. 152–161. DOI: 10.1145/1869790.1869815.

[5] Christian Arnold. "Structure from Motion with Planar Homography Estimation". PhD thesis. Air Force Institute of Technology, 2019.

[6] General Atomics. *General Atomics Legacy GCS.pdf*. 2014. URL: http://www.ga-asi.com/legacy-gcs (visited on 03/03/2019).

[7] General Atomics. *General Atomics Lynx Radar*. 2015. URL: http://www.ga-asi.com/lynx-multi-mode-radar (visited on 03/03/2019).

[8] General Atomics. *RPA 5 million flight hours - 2018.pdf*. 2018. URL: http://www.ga-asi.com/predator-series-aircraft-pass-five-million-flight-hour-mark.

[9] Gill Barequet and Sariel Har-Peled. "Efficiently Approximating the Minimum-Volume Bounding Box of a Point Set in Three Dimensions". In: *Journal of Algorithms* 38.1 (2001), pp. 91–109. ISSN: 01966774. DOI: 10.1006/jagm.2000.1127.

[10] Eivind Bøhn et al. "Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization". In: *CoRR* abs/1911.05478 (2019). arXiv: 1911.05478. URL: http://arxiv.org/abs/1911.05478.

[11] Kendra L.B. Cook. "The silent force multiplier: The history and role of UAVs in warfare". In: *IEEE Aerospace Conference Proceedings* (2007). ISSN: 1095323X. DOI: 10.1109/AERO.2007.352737.

[12] Chris Crispin and Andras Sobester. "An intelligent, heuristic path planner for multiple agent unmanned air systems". In: *AIAA SciTech 56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* January (2015). ISSN: 00207292. DOI: 10.2514/6.2015-0361. URL: http://eprints.soton.ac.uk/374045/.

[13] Willian Dallman. "Infrared and Electro-Optical Stereo Vision for Automated Aerial Refueling". PhD thesis. Air Force Institute of Technology, 2019.

[14] Jean Ponce David A. Forsyth. *Computer Vision: A Modern Approach*. 2nd. PEarson, 2010, pp. 345–392. ISBN: 978-0136085928. URL: http://cmuems.com/excap/readings/forsyth-ponce-computer-vision-a-modern-approach.pdf.

[15] Department of Defense. *Joint Fire Support 3-09*. 2014.

[16] Hongli Deng et al. "Principal curvature-based region detector for object recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2007). ISSN: 10636919. DOI: 10.1109/CVPR.2007.382972.

[17] Department of Defense. *Joint operations 3-0*. 2017, p. 60. ISBN: 3-00. DOI: 10.4135/9781412952446. arXiv: 1403.1385.

[18] Department of Defense. *Joint Targeting 3-60*. January. 2013, p. 137.

[19] Debidatta Dwibedi. "Deep Cuboid Detection: Beyond 2D Bounding Boxes". In: (2016). arXiv: arXiv:1611.10010v1. URL: https://arxiv.org/abs/1611.10010.

[20] Benjamin M. Fain. “Air force institute of technology”. PhD thesis. Air Force Institute of Technology, 2017, p. 120.

[21] Sergio Garc and Science Thesis Stockholm. “Fitting Primitive Shapes to Point Clouds for Robotic Grasping Master of Science Thesis Fitting Primitive Shapes to Point Clouds for Robotic Grasping”. In: (2009). ISSN: 1076-0512. DOI: ISSN-1653-5715. URL: https://pdfs.semanticscholar.org/209b/c5f0c6e9c9230a8e88ed213bea73pdf.

[22] General Atomics. “SKYGUARDIAN SKYGUARDIAN Civilian Airspace Compliant Civilian Airspace Compliant”. In: (2017). URL: www.ga-asi.com.

[23] General Atomics - Aeronautical Systems Inc. “MQ-9 Reaper/Predator”. In: R 041514 (2015), p. 2. URL: http://www.ga-asi.com/Websites/gaasi/images/products/aircraft{_}systems/pdf/MQ9Reaper{_}Predator{_}B{_}032515.pdf.

[24] Rafael C. Gonzalez. *Digital Image Processing*. Ed. by Richard Woods. 2nd. University of Tennessee: Prentice Hall, 2002. URL: http://web.ipac.caltech.edu/staff/fmasci/home/astro{_}refs/Digital{_}Image{_}Processing{_}2ndEd.pdf.

[25] J W Goodman. *Statistical Optics*. A Wiley-Interscience publication. Wiley, 1985. ISBN: 9780471015024. URL: https://books.google.com/books?id=2VTwAAAAMAAJ.

[26] C. Harris and M. Stephens. “A Combined Corner and Edge Detector”. In: (2013), pp. 23.1–23.6. ISSN: 09639292. DOI: 10.5244/c.2.23. arXiv: 0804.1469. URL: http://www.bmva.org/bmvc/1988/avc-88-023.pdf.

[27] Test Image. “Canny Edge Detection - Walk Through”. In: (2009), pp. 1–7. URL: http://www.cse.iitd.ernet.in/{~}pkalra/col783-2017/canny.pdf.

[28] T. Pajdla J. Matas, O. Chum, M. Urban. *Robust Wide Baseline Stereo from Maximally Stable Extremal Regions*. Tech. rep. 2002. DOI: 10.1016/b978-0-12-803662-4.00006-0. URL: http://cmp.felk.cvut.cz/{~}matas/papers/matas-bmvc02.pdf.

[29] Daniel Cremers Jakob Engel, Thomas Schops. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *Technical University Munich* (2016). URL: https://vision.in.tum.de/{_}media/spezial/bib/engel14eccv.pdf.

[30] Gareth James et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.

[31] Ryan Kilgore et al. "Mission Planning and Monitoring for Heterogeneous Unmanned Vehicle Teams: A Human-Centered Perspective". In: *AIAA Infotech@Aerospace 2007 Conference and Exhibit* May (2007), pp. 1–17. DOI: 10.2514/6.2007-2788. URL: http://arc.aiaa.org/doi/10.2514/6.2007-2788.

[32] Reinhard Klette. *Concise Computer Vision*. 2014, p. 441. ISBN: 978-1-4471-6319-0. DOI: 10.1007/978-1-4471-6320-6. URL: http://link.springer.com/10.1007/978-1-4471-6320-6.

[33] Dzenan Lapandic, Jasmin Velagic, and Haris Balta. "Framework for automated reconstruction of 3D model from multiple 2D aerial images". In: *Proceedings Elmar - International Symposium Electronics in Marine* 2017-Septe.September (2017), pp. 173–176. ISSN: 13342630. DOI: 10.23919/ELMAR.2017.8124461.

[34] MAJOR LENDRICK Y. JAMES. "OF ALLOWING ENLISTED PERSONNEL TO FLY MEDIUM-". PhD thesis. South Carolina State, 2016.

[35] Matt Leotta. "TeleSculptor". 2021. URL: https://github.com/Kitware/TeleSculptor.

[36] Weibo Li, Yudong Qi, and Jiaxin Lin. "The research of the aircraft flight path planning under uncertain environment". In: *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference* (2014), pp. 874–878. DOI: 10.1109/CGNCC.2014.7007324. URL: http://ieeexplore.ieee.org/document/7007324/.

[37] Stephan Losey. *The emerging soft tissue paradigm in orthodontic diagnosis and treatment planning* . 2018. URL: https://www.airforcetimes.com/news/your-air-force/2018/03/05/fewer-planes-are-ready-to-fly-air-force-mission-capable-rates-decline-amid-pilot-crisis/.

[38] David Lowe. *Computer Vision*. Tech. rep. Washington State. DOI: 10.1109/T-ED.1980.20172. URL: https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect6.pdf.

[39] Signal Magazine. *USAF acquires 36 Reapers*. 2017. URL: https://www.afcea.org/content/Blog-us-air-force-procures-36-mq-9-reapers.

[40] Agostino Martinelli et al. "A comparison of line extraction algorithms using 2D range data for indoor mobile robotics". In: *Autonomous Robots* 23.2 (2007), pp. 97–111. ISSN: 0929-5593. DOI: 10.1007/s10514-007-9034-y. URL: https://infoscience.epfl.ch/record/97571/files/nguyen{_}2005{_}a{_}comparison{_}of.pdf;.

[41] Lance Menthe, Myron Hura, and Carl Rhodes. *The Effectiveness of Remotely Piloted Aircraft in a Permissive Hunter-Killer Scenario*. Tech. rep. RAND, 2014. URL: https://www.rand.org/content/dam/rand/pubs/research{_}reports/RR200/RR276/RAND{_}RR276.pdf.

[42] National Geospatial Agency. "National Geospatial-Intelligence Agency (Nga) Standardization Document Universal Grids". In: (2014). URL: http://earth-

info.nga.mil/GandG/publications/NGA{_}STND{_}0037{_}2{_}0{_}0{_}GRIDS/NGA.STND.0037{_}2.0.0{_}GRIDS.pdf.

[43] Qiong Nie, Samia Bouchafa, and Alain Merigot. "Voting spaces cooperation for 3D plane detection from monocular image sequences". In: *2012 3rd International Conference on Image Processing Theory, Tools and Applications, IPTA 2012* (2012), pp. 135–140. DOI: 10.1109/IPTA.2012.6469547.

[44] Robert J Noll. "Zernike polynomials and atmospheric turbulence". In: *JOsA* 66.3 (1976), pp. 207–211.

[45] Scott (AFIT) Nykl. *AftrBurner*. Wright Patterson AFB, 2019. URL: https://git.nykl.net.

[46] Ryan T. Olson. "Flight Test Evaluation of Pilot Control Interfaces for Remotely Piloted Vehicles". In: *AIAA Atmospheric Flight Mechanics Conference* June (2015), pp. 1–10. DOI: 10.2514/6.2015-2397. URL: http://arc.aiaa.org/doi/10.2514/6.2015-2397.

[47] OpenCV. *OpenCV: Real Time pose estimation of a textured object*. 2019. URL: http://docs.opencv.org/3.0.0/dc/d2c/tutorial{_}real{_}time{_}pose.html.

[48] Isil Oz, Haluk Rahmi Topcuoglu, and Murat Ermis. "A meta-heuristic based three-dimensional path planning environment for unmanned aerial vehicles". In: *Simulation* 89.8 (2013), pp. 903–920. ISSN: 17413133. DOI: 10.1177/0037549712456419.

[49] Trupti P Patel and Sandip R Panchal. "Corner Detection Techniques : An Introductory Survey". In: *International Journal of Engineering Development and Research* 2.4 (2014), pp. 3680–3686.

[50] Oriana Pawlyk. *Air Force Sets Goal of 20 Flight Hours Per Month for Pilots*. 2018. URL: https://www.military.com/daily-news/2018/03/14/air-force-sets-goal-20-flight-hours-month-pilots.html.

[51] Oriana Pawlyk. *More RPAs than any other pilot position*. 2018. URL: https://www.military.com/daily-news/2017/03/08/drone-milestone-more-rpa-jobs-any-other-pilot-position.html (visited on 02/01/2019).

[52] Leelavathy S R. "Providing Localization using Triangulation Method in Wireless Sensor Networks". In: *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* 6 (2014), pp. 2278–3075. DOI: ISSN:2278-3075. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.678.8860{\&}rep=rep1{\&}type=pdf.

[53] Raytheon. *ANDAS-4 Press Release.pdf*. 2016. URL: http://investor.raytheon.com/phoenix.zhtml?c=84193{\&}p=irol-newsArticle{\&}ID=2163329 (visited on 03/03/2019).

[54] Maj. Reynolds, Justin. *Multi-Domain C2.pdf*. 2018.

[55] Edward Rosten and Tom Drummond. "Machine Learning for High-Speed Corner Detection BT - Genetic Programming". In: *Genetic Programming* 3951.Chapter 34 (2006), pp. 430–443. ISSN: 16113349. DOI: 10.1007/11744023_34. URL: http://link.springer.com/chapter/10.1007{\%}2F11744023{_}34{\%}5Cnpapers3://publication/doi/10.1007/11744023{_}34.

[56] Stuart Russell and Peter Norvig. *Aritficial Intelligence - A Modern Approacj*. 3rd Editio. Pearson, 2010, pp. 17–19, 147. ISBN: 9780136042594.

[57] USAF Safety. *F-16 FLIGHT MISHAP HISTORY*. Tech. rep. 2019, p. 1. URL: https://www.safety.af.mil/Portals/71/documents/Aviation/AircraftStatistics/F-16.pdf.

[58] Sanjana C Shekar and D J Ravi. "Image Enhancement and Compression using Edge Detection Technique". In: *International Research Journal of Engineering and Technology(IRJET)* 4.5 (2017), pp. 1013–1017. URL: https://www.irjet.net/archives/V4/i5/IRJET-V4I5342.pdf.

[59] Gilles Simon and Marie Odile Berger. "Pose estimation for planar structures". In: *IEEE Computer Graphics and Applications* 22.6 (2002), pp. 46–53. ISSN: 02721716. DOI: 10.1109/MCG.2002.1046628.

[60] Irwin Sobel. "An Isotropic 3x3 Image Gradient Operator". In: February 2014 (2015).

[61] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[62] Mark Thompson. *Costly Flight Hours*. 2013. URL: http://nation.time.com/2013/04/02/costly-flight-hours/ (visited on 02/01/2018).

[63] Sebastian Thurn. *Probabilistic Robotics*. 1st. 1st. Stanford University, 2005. URL: http://www.probabilistic-robotics.org/.

[64] Map Tools. *MGRS Quick Guide.pdf*. 2019. URL: https://www.maptools.com/tutorials/mgrs/quick{_}guide (visited on 03/03/2019).

[65] USAF. "Joint Publication 3-52". In: November (2014).

[66] Junhao Xiao, Jianhua Zhang, and Jianwei Zhang. "Fast Plane Detection for SLAM from Noisy Range Images in Both Structured and Unstructured Environments". 2014. URL: https://tams-www.informatik.uni-hamburg.de/people/alumni/xiao/publications/Xiao{_}ICMA2011.pdf.

[67] Xuming He, Richard S. Zemel, and Miguel Á. Carreira-Perpiñán. "Multiscale conditional random fields for image labeling". In: *Proceedings of the 2004 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. (2004), pp. 695–702. DOI: 10.1109/CVPR.2004.1315232.

[68] Artem Babenko Yandex and Victor Lempitsky. "Aggregating local deep features for image retrieval". In: *Proceedings of the IEEE International Conference on Computer Vision* 2015 Inter (2015), pp. 1269–1277. ISSN: 15505499. DOI: 10.1109/ICCV.2015.150. arXiv: arXiv:1510.07493v1.

[69] Shichao Yang et al. "Pop-up SLAM: Semantic Monocular Plane SLAM". 2014. URL: http://www.cs.cmu.edu/{~}kaess/pub/Yang16iros.pdf.

[70] Ozgur Yilmaz and Fatih Karakus. "Stereo and kinect fusion for continuous 3D reconstruction and visual odometry". In: 2013, pp. 115–118. ISBN: 978-1-4799-3343-3. DOI: 10.1109/ICECCO.2013.6718242. URL: https://www.researchgate.net/publication/269327935{_}Stereo{_}and{_}kinect{_}fusion{_}for{_}continuous{_}3D{_}reconstruction{_}and{_}visual{_}odometry.

[71] Zhengyou Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on pattern analysis and machine intelligence* 22 (2000).

[72] Zhiaoliang Zheng. "Point Cloud-Based Target-Oriented 3D Path Planning for UAVs". PhD thesis. UC San Diego, 2019.

Appendix A. Joint Targeting Cycle

A.1 Joint Publications Overview

The Joint Publications (JP) are published by the Joint Chiefs of Staff. They are designed to provide a framework for all military operations. Specifically, they outline high-level strategic doctrine and then translate that doctrine into the operational planning and tactical execution while avoiding specific implementation. This flow can been seen in the concept of "unified action" in *Figure 89*. The Joint Publications are divided into 6 major sections: 1) Personnel, 2) Intelligence, 3) Operations, 4) Logistics, 5) Planning, and 6) Communication Systems. Each of these sections has a primary reference appended with -0. These -0 publications define the major section broadly and tie all of the other publications together in the major section. This thesis will focus on 3-0 "Joint Operations" and associated publications.

The JP 3-0 addresses a number of things relevant to this thesis. First, it shows roughly (there are DoD publications outside of the Joint Publications that do this more specifically) how political policy and philosophy is translated into strategic doctrine. Reference *Figure 89* again to see the parties involved at each step. As stated earlier, it also broadly defines the transition from operational to tactical. But, in order to get more specific, JP 3-60 "Joint Targeting" must be referenced. JP 3-60 further defines the processes at the operational and tactical level. The operational planning cycle is first defined with a template known as the Joint Targeting Cycle. When instantiated for the air power component specifically, it becomes the Joint Air Tasking Cycle. And within the Joint Air Tasking Cycle, one of the steps is further expanded into tactical execution of the plan. This execution step is known as F2T2EA (Find, Fix, Track, Target, Engage, Assess) or the "Kill Chain" [18].

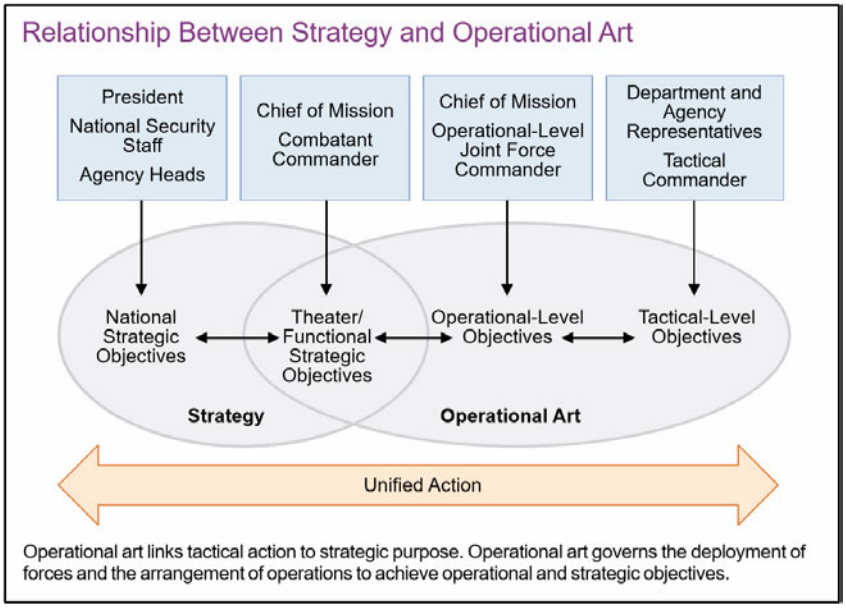

Figure 89: *Strategic, Operational, and Tactical Levels [18]*

The first process that is necessary to understand is the Joint Targeting Cycle (JTC). From the JP 3-60:

> The Joint Targeting Cycle supports the JFCs joint operation planning and execution with a comprehensive, iterative, and logical methodology for employing the ways and means to create desired effects that support achievement of objectives ... The process is not time-constrained nor rigidly sequential. Steps may occur concurrently, but it provides an essential framework to describe the steps that must be satisfied to conduct joint targeting successfully. The deliberate and dynamic nature of the joint targeting cycle supports joint operation planning and execution, providing the depth and flexibility required to support the CONOPS (CONcept of OPerationS) and commanders intent as opportunities arise and plans change [18].

The JTC represents a template for the Joint Force Commander (JFC) to continuously plan daily. In *Figure 90* the six steps of the JTC can be seen:

End State and Commander's Objectives are translated from higher level strategic objectives and from operational imperatives as a result of mission execution.

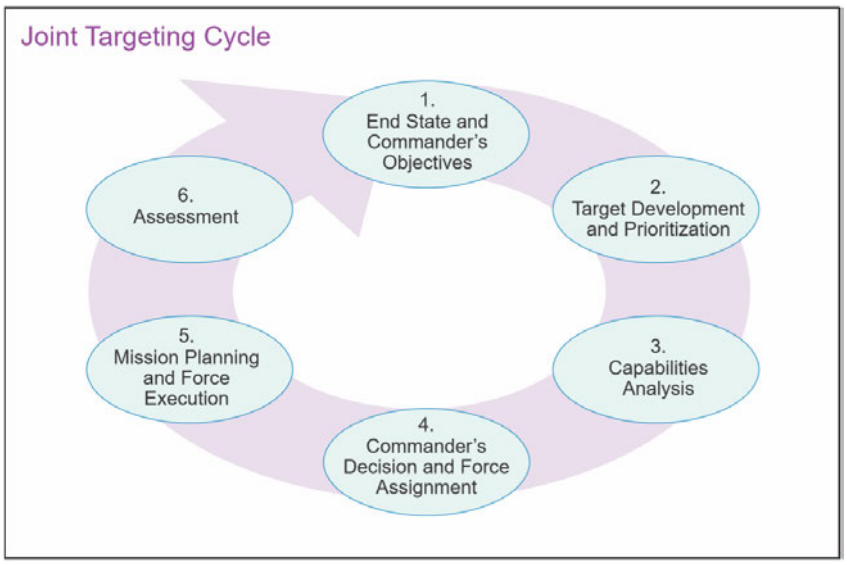

Figure 90: *Joint Targeting Cycle [18]*

Target Development and Prioritization is translating those high level objectives into specific targets with assigned importance levels.

Capabilities Analysis surveys what assets are present and what they are capable of doing.

Commander's Decision and Force Assignment is where targets are paired with assets and assigned missions to execute.

Mission Planning and Force Execution is handled at the tactical level and represents the pre-mission planning and execution accomplished by the operator.

Assessment passes direct mission effects back up the command chain where the results are evaluated in the larger operational picture.

While the JTC is useful conceptually, it lacks specificity necessary for implementation. This specificity is derived through an air component instantiation of the JTC.

A.2 Joint Air Tasking Cycle and Battle Tracking

The Joint Air Tasking Cycle (JTAC) represents the JTC translated into air component specific operations planning. In *Figure 91* it can be seen that there are slight differences. Step 3 - "Capabilities Analysis has been replaced" with "Weaponeering and Allocation". This is because the only assets are aircraft. Thus, the decision is now what aircraft goes with which target and what weapons or payloads it will carry. Step 4 - "Commander's Decision and Force Assignment" has been replaced with "ATO Production and Dissemination". This change comes from the fact that number of aircraft is relatively small and their scope of operation is broad. Thus, their operations can be centrally allocated with standardized and predictable control documents. In this same line of reasoning, the data transfer protocols between each step in the process are also specified. This is important to automation because it defines expected start states, end states, and data input/output protocols and expectations.

It is important to understand the idea of an Air Tasking Order (ATO) and Airspace Control Order (ACO) [65] as they will be simulated later. They are the end product of the first four steps of the Joint Air Tasking Cycle leading up op to Force Execution. There are three main things to notice from *Figure 92*. First, the cycle has 72 hours of lead up, which demonstrates the massive amount of planning that occurs and the huge opportunity for automation to revolutionize the process [18]. Second, these cycles are overlapping 24 hour periods with data feeding in real time both ways with current mission execution and future planning. Third, the ATO is what tells a pilot roughly what they are going after (critical to start the find phase) and the ACO tells them where they should go and where they can not go (based on other aircraft airspace and restrictions) [65].

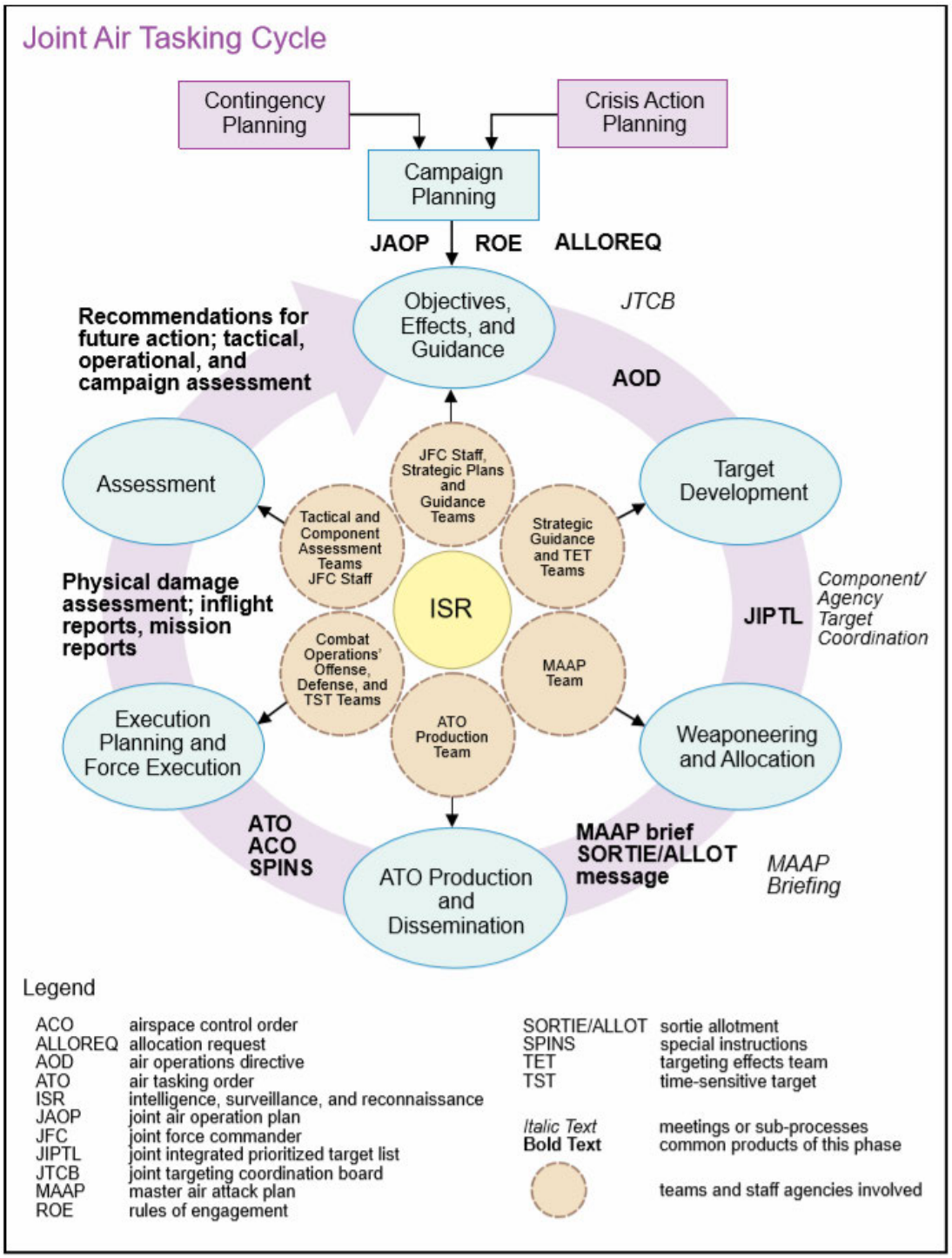

Figure 91: *Joint Air Tasking Cycle [18]*

Additionally, it is necessary to cover the idea of "Battle Tracking". From the JP 3.09:

> Battle tracking is the process of building and maintaining an overall picture of the operational environment that is accurate, timely, and relevant. Effective battle tracking increases the probability of CAS [Close Air Support] attack success by ensuring its application at the proper time and place. The level of detail required and scope of the picture will depend on the mission and information requirements of the joint force. At the tactical level, the simplest form of battle tracking is the mental and graphic picture built and maintained by using maps, observations, and battle updates from C2. At higher levels, battle tracking is more complex and takes advantage of digital information systems using multiple sources to generate a coherent picture of the operational environment. Effective battle tracking will aid in maintaining an understanding of friendly and enemy progress, reduce redundant targeting, and reduce the possibility of friendly fire. Effective methods of battle tracking include maintaining up-to-date maps, imagery, and status boards, and utilizing computerized tracking and display methods [15].

This concept, combined with the JTAC, produces the operational framework surrounding the tactical execution and prototypes inputs and outputs for automation.

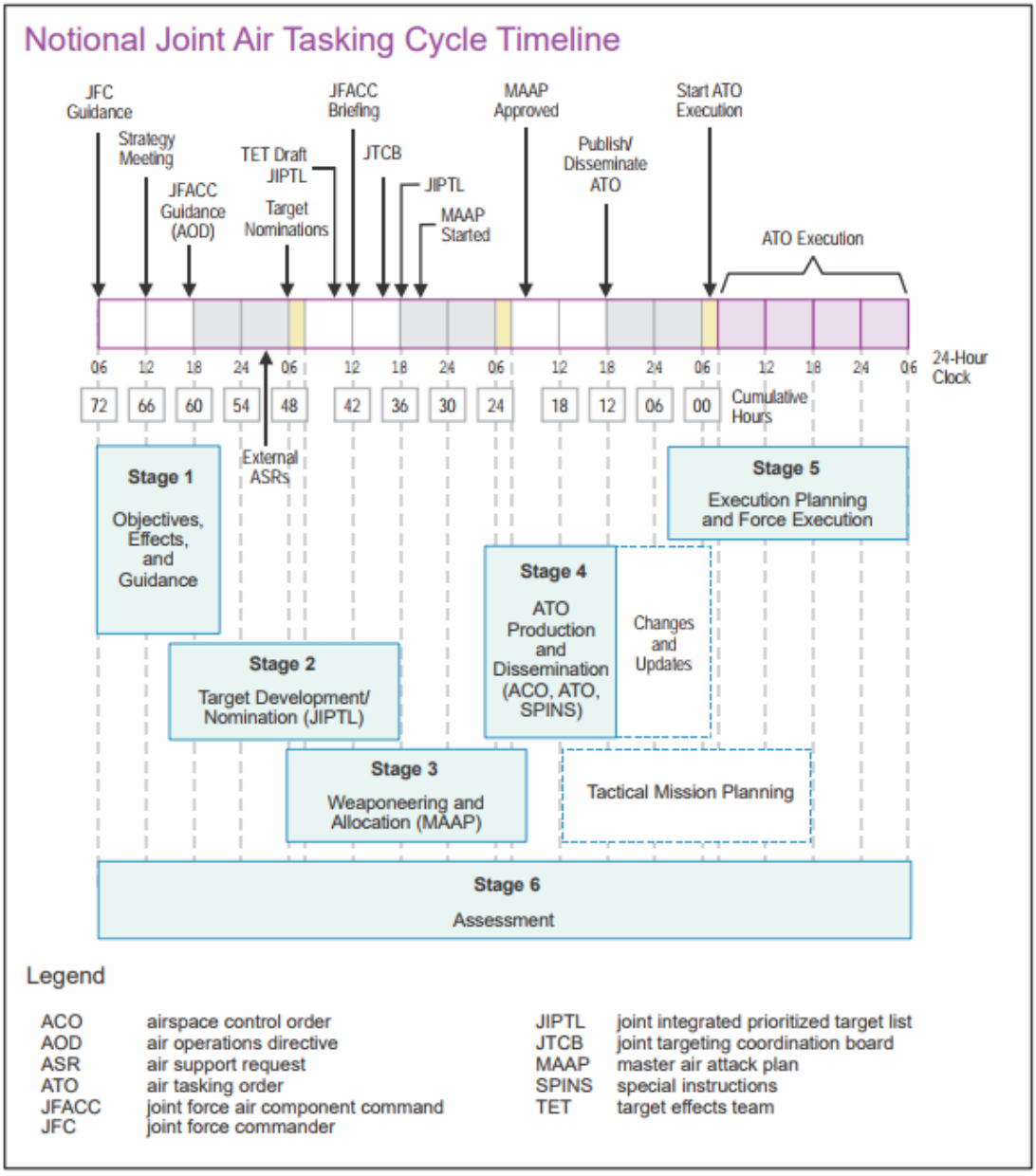

Figure 92: *The Joint Air Tasking Cycle - A time-line and physical products derived from the Joint Targeting Cycle [18]*

Appendix B. Installing the software

B.1 Installing and running the simulation

```
## Automated Find Fix and TRACK (AFFTRAC)

### Setup Instructions

 1. First you will need access to Dr Scott Nykl's **AFTR Burner**
 graphics engine (Not publicly available at this time)

 2. In the **/usr** folder of AFTR Burner, **Clone** this git
 repo to that location

 3. Run the **installOCV.sh** script found in **/opencv** to
 download and install OpenCV with the extra-modules (if it fails,
 attempt to run it a second time, or if \*.sh files are not
 recognized, download Git Bash and try again)

 4. Ensure **BOOST** system variables are configured properly
 (the configuration is identical to AFTR Burner's)

 5. Create a new **System Variable** called "**OpenCV_DIR**"
 and set it to the location of the newly created **/opencv/Install**
 folder

 6. Run the **BUILD 64 BIT MSVC 2017.bat** script, click
 **Configure**, then **Generate**, and then **Open Project**
```

```
 7. Once the project is open in Visual Studio, right-click on
 **AFFTRACK** and select **Set as StartUp Project**

 8. Finally, change the **Solutions Configurations** under
 toolbar to **MinSizeRel** and build and compile the project
 by clicking **Local Windows Debugger** or pressing **CTRL + F5**
 (alternatively, you may change the Solutions Configuration to
 Debug for debugging)

(Note:  Depending on system set-up, during run-time you may get
an error where certain opencv files (opencv_world401.dll for
example) can't be found.  If this happens add a path variable
**ex: \opencv\Install\opencv\x64\vc15\bin** where the file
lives in the openCV install.

 ### Operating Instructions

*The program is still in a state of development thus all key-bindings
are subject to change*

**Quick Start Instructions**

**SIMULATION ONLY**
1. Alter aftr.config file to one of 4 scenarios (set environment
equal to either CityBlock, NotreDame, ArabBuildingSingle, or
```

ArabBuildingMulti).

2. Alter aftr.config file to set fov=2.5 for CityBlock or NotreDame and fov=0.4 for ArabBuildingSingle and fov=0.6 for ArabBuildingMulti. You can use other values and the program will scale automatically but these are good starting points that have a FOV appropiate for the size of building being looked at.

3. Set other variables as desired. aftr.conf has sufficient explanation to understand function.

4. **Wait** for all textures to load once program starts (approximately 1 minute) before doing anything manually. Recommend pressing **p** to freeze the A/C motion as everything happens automatically.

5. Otherwise, press 1 or 3 to follow A/C in and it will automatically assume an orbit, generate all images, perform matching, clean the point cloud, and hold in derived good hold area.

FLIGHT TEST MODE
1. Alter aftr.config file to one of 9 scenarios (set testLocation equal to Simple1,Simple2,etc...).

2. Set configuration variables for flight test or play-back.

a. Flight Test -

   testMode=True

   testLogData=True

   testUpdateHold=False

   logPlayback=False

b. Playback

   testMode=False

   testLogData=False

   testUpdateHold=False

   logPlayback=True

3. Set other variables as desired. aftr.conf has sufficient explanation to understand function.

4. **Wait** for all textures to load once program starts (approximately 1 minute) before doing anything manually. Recommend pressing **p** to freeze the A/C motion as everything happens automatically.

5. Otherwise, press 3 to assume God's eye view and watch log file play back or conduct test.

Controls

* **Mouse Controls**

   - **Left-click**: Look around

```
   - **Right-click**: Move in look direction
   - **Scroll-wheel**: Increase or decrease movement speed


* **Keyboard Controls**
   - **1**: Toggle between chase camera for drone and
            "free" camera
   - **2**: Focus on Target
   - **3**: Gods eye view
   - **4**: Toggle point cloud visibility
   - **5**: Toggle truth object visibility (DEPRECATED)
   - **7**: Manually trim outliers (DEPRECATED)
   - **8**: Manually regenerate probability cloud
            (good hold area)

   - **C**: Generate camera calibration data (necessary if
            you reload from parameters)
   - **M**: Set camera on "demo mode" for external screen
            capture software
   - **N**: Turn off "demo mode"
   - **P**: Pause aircraft motion if able
   - **H**: "Hide" target from algorithm so it thinks it
            can't find it.
   - **W**: Start/stop the dummy target walking to the next
            purple waypoint
   - **R**: Stop the dummy target and move to waypoint 0
```

- **Period(.)**: Cycle through images doing point matching
  to generate point cloud points

- **Ctrl-S** Save out current point cloud to .csv file in
  current working directory

- **F10** Take screenshot

Appendix C. Planned Test Locations and Scenarios

Table 10: *Path Coordinates 1*

Profile	**Point**	**Time (min)**	**Latitude**	Longitude
S1_A	S1_A1	0:00	32.38423496	-106.4832488
	S1_A2	4:00	32.38430061	-106.4832505
	S1_A3	8:00	32.38430271	-106.4833513
S2_A	S2_A3	0:00	32.38458769	-106.4878575
	S2_A2	4:00	32.38458545	-106.4879768
	S2_A1	8:00	32.38450385	-106.4879757
S2_B	S2_B1	0:00	32.38436998	-106.4875884
	S2_B2	2:30	32.38446425	-106.4875898
	S2_B3	5:00	32.38453984	-106.4875911
	S2_B4	7:30	32.38461476	-106.4875908
I1_A	I1_A3	0:00	32.37914146	-106.4890918
	I1_A2	4:00	32.37927088	-106.4892561
	I1_A1	8:00	32.37913036	-106.4894461
I1_B	I1_B1	0:00	32.37867693	-106.4887558
	I1_B2	2:30	32.37876687	-106.4885058
	I1_B3	5:00	32.37926225	-106.4885482
	I1_B4	7:30	32.37911802	-106.4890384

Table 11: *Path Coordinates 2*

Profile	**Point**	**Time (min)**	**Latitude**	Longitude
I2_A	I2_A1	0:00	32.37689902	-106.4800389
	I2_A2	4:00	32.37717413	-106.4800367
	I2_A3	8:00	32.37717152	-106.480183
I2_B	I2_B3	0:00	32.3771802	-106.4806333
	I2_B2	2:00	32.37698791	-106.4804006
	I2_B1	4:00	32.37690495	-106.4802844
	I2_B2	6:00	32.37698791	-106.4804006
	I2_B3	8:00	32.3771802	-106.4806333
C1_A	C1_A3	0:00	32.38292806	-106.4810556
	C1_A2	4:00	32.38292793	-106.4808993
	C1_A1	8:00	32.38352598	-106.4809052
C1_B	C1_B1	0:00	32.38275316	-106.481616
	C1_B2	2:00	32.38309273	-106.4813666
	C1_B3	4:00	32.38353495	-106.4814291
	C1_B4	6:00	32.38377094	-106.4814053
	C1_B5	8:00	32.38376853	-106.4812658
C1_C	C1_C1	0:00	32.38292595	-106.4819638
	C1_C2	2:30	32.38349057	-106.4819793
	C1_C3	5:00	32.38351334	-106.481767
	C1_C4	7:30	32.38375793	-106.4817759
C2_A	C2_A1	0:00	32.3824279	-106.4853724
	C2_A2	4:00	32.38260685	-106.4852357
	C2_A3	8:00	32.38277134	-106.4852403
C2_B	C2_B1	0:00	32.38208785	-106.4854561
	C2_B2	2:00	32.38208649	-106.4857747
	C2_B3	4:00	32.3820911	-106.4860209
	C2_B4	6:00	32.38223515	-106.4860872
	C2_B5	8:00	32.3826074	-106.486082

Table 12: *C-12J Planned Sortie Profiles*

Aircraft	**Run**	**Environment**	**Maneuver**	**Target Profile**	**Sortie**
C-12J	1	Simple 1	4.5km 360 Orbit	Stationary	A1
	2	Simple 1	6.0km 360 Orbit	Moving	A1
	3	Simple 1	6.0km 360 Orbit	SOI	A1
	4	Simple 1	7.5km 360 Orbit	Stationary	A1
	5	Simple 1	Cloverleaf	Stationary	A1
	6	Simple 1	AFFTRAC	Stationary	A1
	7	Simple 1	AFFTRAC	Moving	A2
	8	Simple 1	AFFTRAC	SOI	A2
	9	Intermediate 1	6.0km 360 Orbit	Moving	A2
	10	Intermediate 1	6.0km 360 Orbit	SOI	A2
	11	Intermediate 1	AFFTRAC	Stationary	A2
	12	Intermediate 1	AFFTRAC	Moving	A2
	13	Intermediate 1	AFFTRAC	SOI	A2
	14	Complex 1	6.0km 360 Orbit	Moving	A3
	15	Complex 1	6.0km 360 Orbit	SOI	A3
	16	Complex 1	AFFTRAC	Stationary	A3
	17	Complex 1	AFFTRAC	Moving	A3
	18	Complex 1	AFFTRAC	SOI	A3

Table 13: *MQ-9 Planned Sorties Profiles*

Aircraft	**Run**	**Environment**	**Maneuver**	**Target Profile**	**Sortie**
MQ-9	19	Simple 1	4.5km 360 Orbit	Stationary	B1
	20	Simple 1	6.0km 360 Orbit	Moving	B1
	21	Simple 1	6.0km 360 Orbit	SOI	B1
	22	Simple 1	7.5km 360 Orbit	Stationary	B1
	23	Simple 1	Cloverleaf	Stationary	B1
	24	Simple 1	AFFTRAC	Stationary	B1
	25	Simple 1	AFFTRAC	Moving	B1
	26	Simple 1	AFFTRAC	SOI	B1
	27	Simple 1	MQ-9 IP	Stationary	B1
	28	Simple 1	MQ-9 IP	Moving	B1
	29	Simple 1	MQ-9 IP	SOI	B1
	30	Intermediate 1	4.5km 360 Orbit	Stationary	B2
	31	Intermediate 1	6.0km 360 Orbit	Moving	B2
	32	Intermediate 1	6.0km 360 Orbit	SOI	B2
	33	Intermediate 1	7.5km 360 Orbit	Stationary	B2
	34	Intermediate 1	AFFTRAC	Stationary	B2
	35	Intermediate 1	AFFTRAC	Moving	B2
	36	Intermediate 1	AFFTRAC	SOI	B2
	37	Intermediate 1	MQ-9 IP	Stationary	B2
	38	Intermediate 1	MQ-9 IP	Moving	B2
	39	Intermediate 1	MQ-9 IP	SOI	B2
	40	Complex 1	4.5km 360 Orbit	Stationary	B3
	41	Complex 1	6.0km 360 Orbit	Moving	B3
	42	Complex 1	6.0km 360 Orbit	SOI	B3
	43	Complex 1	7.5km 360 Orbit	Stationary	B3
	44	Complex 1	AFFTRAC	Stationary	B3
	45	Complex 1	AFFTRAC	Moving	B3
	46	Complex 1	AFFTRAC	SOI	B3
	47	Complex 1	MQ-9 IP	Stationary	B3
	48	Complex 1	MQ-9 IP	Moving	B3
	49	Complex 1	MQ-9 IP	SOI	B3

Appendix D. In Depth Data Extraction and Analysis

D.1 Reaper Analysis Toolkit (RAT)

The Reaper Analysis Toolkit was a software package which enabled efficient extraction and examination of MQ-9 log data. It aggregated all of the collected flight and systems parameters from a given sortie into a time-tagged, easy to consume .csv log file. RAT was written in the Python programming language, version 2.7. It was created by Dr. Michael Grimaila at the Air Force Institute of Technology. The original source code was modified to use Python 3.0 for this analysis.

D.2 Holding area density

Data analysis for creating the holding area density from MQ-9 Logs file followed the process listed below:

1. Pull the video and log files from the MQ-9 GCS
2. Process the telemetry into useable formats with the RAT
3. Combine relevant video files into continuous video files
4. Correlate the video to the correct timestamp in MQ-9 telemetry
5. Reduce the MQ-9 telemetry to a form AFFTRAC can consume
6. Select images from the continuous video file at a regular frame rate interval
7. Manually determine target location, enter the pixel values into the MQ-9 telemetry file
8. Run AFFTRAC on video, telemetry, and processed images to produce a new log file

9. Process the AFFTRAC log file in MATLAB

10. Conduct statistical analysis on aggregate MATLAB results

Pull the video and log files from the MQ-9 GCS.

After the last MQ-9 sortie of the day (i.e. the last seat swap), all of the MQ-9 video and telemetry was downloaded from the MQ-9 multi-function workstations (MFW) aft of the pilot sensor control racks in the GCS. As each of the MFWs were Microsoft Windows workstations with a DVD burner, a series of DVDs were burned to accomplish this.

Process the telemetry into useable formats with the RAT.

As described previously, the RAT tool was capable of converting raw MQ-9 telemetry data into easily useable .csv files. Only the subsequent M telemetry files during the appropriate time of the test points were used.

Combine relevant video files into continuous video files.

The MQ-9 GCS is only capable of storing 15 minutes of continuous video in a single file. This necessitating combining of disparate video clips when the test point was not contained within a single clip. Free software video editor Shotcut was used to combine disparate video files.

Correlate the video to the correct timestamp in MQ-9 telemetry.

The subsequent video files next had to be precisely matched to the correct video telemetry. Prior to every test point, destructive graphics were cycled on and off for just this reason. This allowed comparison with timestamps present on secondary video streams. This was further confirmed by obvious visual references. By this

process, the 1Hz telemetry was matched to the appropriate point in the video that represented the start of each test point.

Reduce the MQ-9 telemetry to a form AFFTRAC can consume .

The M files from the RAT tool have 2,982 different parameters, but only eight were required for AFFTRAC playback, including target location, GPS time, and a mid-level voting (MLV) aircraft position and heading from the three embedded GPS/INS on the aircraft. An extract of the required parameters in a log file is presented in *Figure 93*.

Select images from the continuous video file at a regular frame rate interval .

A series of frames were pulled from the video at regular intervals. Using the constant airspeed of the aircraft and known frame rate of the video (30 frames per second), an image was pulled every 270 frames or every 9 seconds. This resulted in approximately 45 frames per orbit depending on the radius of the circle flown. This created the required 5-10 degrees of parallax between subsequent images.

Manually determine target location, enter the pixel values into the MQ-9 telemetry file.

AFFTRAC took target location in the environment as an input, so it was necessary to manually define where the target was in each image frame where it was visible. This

	A	B	C	D	E	F	G	H
1	Tgt Gnd Elev	TGT Latitude	TGT Longitude	GPS Time	MLV Lat	MLV Long	MLV Alt MSL-84	MLV Hdg
2	4306	32.37900162	-106.4890594	239882	32.39424133	-106.4283295	25163.42969	329.6058655
3	4306	32.37900162	-106.4890594	239883	32.39424133	-106.4283295	25163.42969	329.6058655
4	4306	32.37900162	-106.4890594	239884	32.39424133	-106.4283295	25163.42969	329.6058655

Figure 93: *Reduced M-File*

was done by adding pixel space coordinates into the MQ-9 telemetry file processed previously. The form is shown in *Figure 94*.

In the first column after aircraft heading, a value of -1 indicates that there was no image key frame to process. A non-negative number (for example, 3) indicates a key frame was present in the file 3.bmp. In the next 4 columns, the target was identified in pixel space from the top left corner of the image if there were values other than -1. If these 4 columns were -1, there was no visible target. The end result was that this allowed AFFTRAC to determine where the target existed in pixel space and thus where it existed in reconstructed 3D space between pairs of images. An example of a modified image is shown in *Figure 95* where the target was demarked with an orange square based on the min/max x/y values from the .csv file.

Run AFFTRAC on video, telemetry, and processed images to produce a new log file.

AFFTRAC was fed the data products from previous steps and allowed to play back the information as if it was receiving all the data real-time. During this playback of a 360 orbit, AFFTRAC determined when subsequent image pairs both contained a designated target. This was then rectified within a partial structural point cloud and used to create a holding area density. The event of creating this holding probability cloud was marked with a flag in the AFFTRAC created log. The end result was

	H	I	J	K	L	M
1	MLV Hdg	Key Frame	Tgt Min X Pixel	Tgt Min Y Pixel	Tgt Max X Pixel	Tgt Max Y Pixel
2	329.6058655	0	-1	-1	-1	-1
3	329.6058655	-1	-1	-1	-1	-1
28	306.7539368	-1	-1	-1	-1	-1
29	306.7539368	3	1010	571	1013	574
30	303.2492676	-1	-1	-1	-1	-1

Figure 94: *Pixel-Space Target Coordinates in a Reduced M-File*

Figure 95: *Derived Target Location (In Orange)*

an AFFTRAC log with holding area density generation information for every pair of images processed. This was then fed into MATLAB as described below.

Process the AFFTRAC log file in MATLAB.

MATLAB was used to process the AFFTRAC log file and analyze each holding area density created during AFFTRACs sequential image analysis. AFFTRAC output a flag for log entries which corresponded to an update to the holding area density ("$radial\_update\_flag$"). Data for each run were filtered using this flag. Additionally, AFFTRAC processed azimuths of acceptable holding area in 10 degree "slices" and output a flag corresponding to the central azimuth in a given slice. For each acceptable azimuth slice identified by AFFTRAC, a probability distribution representing the holding area density was generated as follows:

- Divide total points equally between all acceptable azimuth slices
- Total points generated for the holding cloud: $\sim$10,000

- Define a uniform distribution between +/-5 degrees from acceptable azimuth
- Define a truncated normal distribution between 4.5-7.5km
- $\mu = 6000$ (m)
- $\sigma = 600$ (m)
- Sample angle and radius values from the uniform and normal distributions, respectively, for each point in the AFFTRAC specified holding area

The process described was the same as that used by AFFTRAC when generating holding area densities during operation. Due to the nature of random sampling, the actual point cloud generated was not identical to that generated by AFFTRAC, but was statistically representative. For visualization, points generated which fell into the "truth" region were plotted in blue and points which fell outside were plotted in red.

Conduct statistical analysis on aggregate MATLAB results.

MATLAB produced a .csv log file that showed whether or not an azimuth was determined usable by AFFTRAC or not. Truth data as to which azimuths the target was viewable was then input into the log file based on the video analysis done in the 4.5, 6.0, and 7.5 km rings. With truth data, comparisons between what AFFTRAC believed versus the actual line-of-sight data allowed determination of relative validity of the AFFTRAC produced point clouds. Each azimuth was marked either good or bad depending on whether line-of-sight would be maintained. From this point, total number of AFFTRAC predicted good azimuths were summed and then the number of total correct and incorrect azimuths were compared to determine overall performance of the holding point cloud generation.

D.3 Aircraft Turn Commands

Data analysis for Aircraft Turn Commands in flight test followed the process listed below.

1. Pull the log files from the MQ-9 GCS
2. Process the telemetry into useable formats with the RAT
3. Process the Cloverleaf log file in MATLAB

Pull the video and log files from the MQ-9 GCS.

After the last MQ-9 sortie of the day (i.e. the last seat swap), all of the MQ-9 video and telemetry was downloaded from the MQ-9 MFW aft of the pilot sensor control racks in the GCS. As each of the MFWs were Microsoft Windows workstations with a DVD burner, a series of DVDs were burned to accomplish this.

Process the telemetry into useable formats with the RAT.

As described previously, the RAT tool was capable of converting raw MQ-9 telemetry data into easily useable .csv files. Only the subsequent M telemetry files during the appropriate time of the test points were used.

Process the Cloverleaf log file in MATLAB.

MATLAB was used to process the Cloverleaf log file. Aircraft position and heading data were taken from the MQ-9 telemetry logs and plotted relative to the position of the holding area density. AFFTRAC-commanded turn directions were taken from the AFFTRAC logs and plotted as arrows along the heading vectors.

Appendix E. Acronyms

- **ACO**: Airspace Control Order
- **AF**: Air Force
- **AFFTRAC**: Automated Find Fix and Track
- **AFB**: Air Force Base
- **AFIT**: Air Force Institute of Technology
- **AOB**: Angle of Bank
- **ATO**: Air Tasking Order
- **ATP**: Advanced Targeting Pod
- **C2**: Command and Control
- **CNN**: Convolutional Neural Net
- **COTS**: Commercial off the shelf
- **DoD**: Department of Defense
- **DRL**: Deep Reinforcement Learning
- **DTIC**: Defense Technical Information Center
- **DVI**: Digital Visual Interface
- **EAR**: Export Administration Regulations
- **EO**: Electro-Optical
- **EO/IR**: Electro-Optical / Infrared

- **ENG**: Department of Engineering (AFIT)
- **ESM**: Electronic Support Measure
- **F2T**: Find, Fix, and Track
- **F2T2EA**: Find, Fix, and Track, Target, Engage, Assess
- **FAST**: Features from Accelerated Segment Test
- **FLTS**: Flight Test Squadron
- **FMV**: Full Motion Video
- **GB**: Gigabyte
- **GBU**: Guided Bomb Unit
- **GCS**: Ground Control Station
- **GPS**: Global Positioning System
- **GMTI**: Ground Moving Target Identification
- **GUI**: Graphical User Interface
- **GWOT**: Global War on Terror
- **HAT**: Height Above Target
- **IMINT**: Image Intelligence
- **INS**: Inertial Navigation System
- **IP**: Instructor Pilot
- **ISR**: Intelligence, Surveillance, and Reconnaissance

- **JFC**: Joint Force Commander
- **JP**: Joint Publication
- **JTAC**: Joint Air Tasking Cycle
- **KCAS**: Knots Calibrated Airspeed
- **KTAS**: Knots True Airspeed
- **MALE**: Medium Altitude Long Endurance
- **MGRS**: Military Grid Reference System
- **MFW**: Multi-Function Workstation
- **MSL**: Mean Sea Level
- **MTS**: Multi-Spectral Targeting System
- **MVBB**: Minimum Volume Bounding Box
- **MWAS**: Maritime Wide Area Surveillance
- **OG**: Operations Group
- **OTF**: Optical Transfer Function
- **PED**: Processing, Exploitation, and Dissemination
- **PID**: Positive Identification
- **RAM**: Random Access Memory
- **RANSAC**: Random Sample Consensus
- **RAT**: Reaper Analysis Toolkit

- **RGB**: Red Blue Green
- **RMSE**: Root Mean Squared error
- **SATCOM**: Satellite Communications
- **SAR**: Synthetic Aperture Radar
- **SfM**: Structure from Motion
- **SIFT**: Scale Invariant Feature Transform
- **SIGINT**: Signals Intelligence
- **SLAM**: Simultaneous Localization and Mapping
- **SOI**: Scenario of Interest
- **SRT**: Standard Rate Turn
- **SUSAN**: Smallest Univalue Segment Assimilating Nucleus
- **TPS**: Test Pilot School
- **TTPs**: Tactic, Technique, and Procedures
- **TW**: Test Wing
- **UAV**: Unmanned Aerial Vehicle
- **U.S.**: United States
- **USAF**: United States Air Force
- **WSMR**: White Sands Missile Range

Vita

Aubrey "BRUISSR" Olson was born in Houston, Texas on January 30th 1986. He is married to his lovely wife and has two young boys and a newborn girl. He graduated from Trinity High School of Midland, Texas in 2004 with honors. He attended Austin College in Sherman, Texas as a Presidential Scholar (full academic merit based scholarship awarded prior to enrollment). Aubrey graduated in the spring of 2008 with dual Bachelor degrees in Computer Science and in Japanese Language and Culture.

Upon graduation he worked as a Logistics Specialist for Raytheon until 2011, where he authored custom AJAX libraries to allow real time metrics dashboards for intra-company inventory analysis. For the next year and a half he worked as a Software Installation Consultant for OpenText. There he analyzed business processes at over half a dozen companies and managed the automation of those human processes with major software installations.

He commissioned into the United States Air Force in October of 2012 as a MQ-9 Remotely Piloted Aircraft (RPA) Pilot. He has flown almost 2,000 hours and was an evaluator pilot. In 2018 he received his first MS in Computational Science and Robotics from the South Dakota School of Mines and Technology. In the fall of 2018, he entered the joint Air Force Institute of Technology and USAF Test Pilot School joint pipeline. He was the first RPA Pilot to do so. He graduated TPS with MS in Aerospace Test Management in December of 2020 and will be awarded his third MS in electrical engineering in March of 2021.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
03–25–2021	Master's Thesis	Sept 2018 — Mar 2021

4. TITLE AND SUBTITLE

AUTOMATED FIND FIX AND TRACK WITH A MEDIUM ALTITUDE LONG ENDURANCE REMOTELY PILOTED AIRCRAFT

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Aubrey L. Olson

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-21-M-069

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

Test Pilot School
220 Wolfe Ave.
Edwards AFB CA 93523
DSN 277-3000, COMM 661-277-3000

10. SPONSOR/MONITOR'S ACRONYM(S)

USAF TPS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A limitation in RPA ISR operations is loss of target track if the command link is severed. For an RPA to effectively execute the ISR mission without a command link, it needs the capability to F2T targets autonomously. Automated Find Fix and Track (AFFTRAC) was developed to help solve this problem by demonstrating a proof of concept tactical autopilot. Monocular stereo vision was used to process sequential images acquired during orbit to produce a partial structural point cloud of the original structure. This partial structural point cloud was then exploited to create a holding area density for the aircraft to stay within. A simple greedy algorithm exploited this holding area density to produce aircraft turn commands to approximate tactical ISR holding. The result was that imagery from existing MQ-9 sensors was used to provide command guidance to autonomously to maintain line of sight to a target. Overall, AFFTRAC is a promising initial framework for a tactical autopilot, but additional development is needed to mature component algorithms.

15. SUBJECT TERMS

ISR, tactical autopilot, disjointed monocular stereo vision, SLAM, SfM, value map, occlusion detection, RPA, UAV, F2T, F2T2EA, kill chain, MQ-9, autonomous, autonomy, greedy algorithm, framework

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Brett Borghetti, AFIT/ENG
U	U	U	U	194	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4612; brett.borghetti@afit.edu