March 2018

Interactive Sensing and Planning

Augustine L. Kelty
Worcester Polytechnic Institute

Maria de los Angeles Sierra Rossi
Worcester Polytechnic Institute

Mary T. Vierling
Worcester Polytechnic Institute

Roger Michael Aiudi
Worcester Polytechnic Institute

Rose Agnes Whittle
Worcester Polytechnic Institute

See next page for additional authors

Follow this and additional works at: https://digitalcommons.wpi.edu/mqp-all

Repository Citation

This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.
RVC1801
Interactive Planning and Sensing
A Major Qualifying Project Report
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science in Aerospace Engineering
By

Roger Aiudi
Augustine Kelty
Mafia Sierra Rossi
Mary Vierling
Rose Whittle

Approved by:

Professor Raghvendra V. Cowlagi, Advisor
Aerospace Engineering Program, WPI

and in Aerospace Engineering and Robotics Engineering
By

Tom Stilwell

Approved by:

Professor Raghvendra V. Cowlagi, Co-Advisor
Aerospace Engineering Program, WPI

Alexander Wyglinski, Co-Advisor
Electrical and Computer Engineering, WPI

March 22, 2018
Abstract

The objective of this project is to demonstrate via hardware experiments the following interactive planning and sensing technique of coordinating the actions of a team of multiple autonomous mobile vehicles. We consider the problem of planning the path of one of these vehicles, designated as the actor, to traverse a threat field with minimum exposure to a prespecified threat. This threat field is an unknown, time-invariant, and strictly positive scalar field defined on a compact 2D spatial domain. The threat field is estimated by measurements taken by the other vehicles in the team, which are designated as sensors. The threat field is assumed to be finitely parametrized by coefficients of spatial basis functions. Estimates of these parameters are constructed using measurements from the sensors. The actor and the sensors interact iteratively to converge upon a sensor placement in the aforesaid 2D domain, as well as a path with minimum expected threat exposure for the actor. In this project, the threat field is represented by an image with pixel color from blue to yellow indicating threat levels. This image is projected onto the ground by multiple projectors inside a large darkened room. The actor vehicle is implemented by a differential-drive wheeled robot, whereas the sensor vehicles are implemented by small electric quadrotors, designed to localize and fly themselves indoors. Downward-pointing cameras on each sensor are calibrated to sense the image intensity or color, and thereby emulate measurement of the threat field. Bench tests are reported for each of the subsystems involved in the overall experimental setup, and the coordinated actions of multiple vehicles is demonstrated via the experiment.
## Authorship Page

<table>
<thead>
<tr>
<th>Section</th>
<th>Author</th>
<th>Project Work</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Background</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Background Introduction</td>
<td>MSR, RW, MV</td>
<td></td>
</tr>
<tr>
<td>Project Objective</td>
<td>MSR, MV, AK</td>
<td></td>
</tr>
<tr>
<td>Related Work</td>
<td>MSR, RA, MV</td>
<td></td>
</tr>
<tr>
<td>IPAS Algorithm</td>
<td>TS</td>
<td></td>
</tr>
<tr>
<td><strong>System Design</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Projection</td>
<td>MV</td>
<td>MV, RA, AK, MSR</td>
</tr>
<tr>
<td>Field Sensing</td>
<td>AK, MV (UAV Design),</td>
<td>AK, RA, RW, MSR</td>
</tr>
<tr>
<td></td>
<td>RA (Sensing Mechanism)</td>
<td></td>
</tr>
<tr>
<td>Localization System</td>
<td>MSR (Selection), MV (Mounts), RA (Usage)</td>
<td>RA, TS</td>
</tr>
<tr>
<td>Path Planning,</td>
<td>RA (Pi 3), TS (IPAS)</td>
<td>TS, RA</td>
</tr>
<tr>
<td>Computations,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>and Communications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path Execution</td>
<td>MV, RW, TS</td>
<td>TS, RW</td>
</tr>
<tr>
<td><strong>System Development</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Projection</td>
<td>MV (Setup), AK (Mounts),</td>
<td>MV, RA, AK, MSR</td>
</tr>
<tr>
<td></td>
<td>RA (Threat Field), MSR (Colormaps)</td>
<td></td>
</tr>
<tr>
<td>UAV Development</td>
<td>AK, MV</td>
<td>AK</td>
</tr>
<tr>
<td>Sensing Mechanism</td>
<td>RA, MSR</td>
<td>RA, MSR</td>
</tr>
<tr>
<td>Localization</td>
<td>MSR (Setup),</td>
<td>RA, TS</td>
</tr>
<tr>
<td></td>
<td>RA (Implementation and Accuracy)</td>
<td></td>
</tr>
<tr>
<td>Implementing IPAS</td>
<td>TS (Relocation), RW (Actor),</td>
<td>TS, RA, RW</td>
</tr>
<tr>
<td></td>
<td>RA (Communications)</td>
<td></td>
</tr>
<tr>
<td>**Results and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conclusions**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Projection</td>
<td>MV (Setup), MSR (Sensors)</td>
<td>MV, RA, AK, MSR</td>
</tr>
<tr>
<td>Pozyx Precision/Accuracy</td>
<td>MSR (Precision), TS (Error), RA (Accuracy)</td>
<td>RA, TS, MSR</td>
</tr>
<tr>
<td>IPAS Algorithm</td>
<td>TS (Relocation and Implementation), RA (Interfacing)</td>
<td>TS, RA</td>
</tr>
<tr>
<td>Localization using</td>
<td>TS</td>
<td>TS</td>
</tr>
<tr>
<td>Wireless Signals</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Contents

1 Background ................................. 7  
1.1 Project Description .......................... 7  
1.2 IPAS Algorithm .............................. 8  
1.3 Related Work .............................. 10  

2 System Design .............................. 13  
2.1 Field Projection ............................ 13  
2.1.1 Threat Field Representation ................. 14  
2.1.2 Projectors ................................ 14  
2.1.3 Mounts and Set Up ......................... 15  
2.2 Field Sensing ............................... 16  
2.2.1 UAV Design ................................ 16  
2.2.2 Sensing Mechanism ......................... 20  
2.3 Localization ............................... 20  
2.3.1 System Selection ......................... 21  
2.3.2 Mounts and Set Up ......................... 22  
2.4 Path Planning, Computations, and Communications .................................. 22  
2.4.1 Raspberry Pi 3 .......................... 22  
2.4.2 IPAS Algorithm .......................... 23  
2.5 Path Execution ............................ 23  
2.5.1 Actor Vehicle ........................... 23  

3 System Development ......................... 25  
3.1 Field Projection ............................ 25  
3.1.1 Projection Set Up ......................... 25  
3.1.2 Projector Mounts ......................... 26  
3.1.3 Threat Field Representation ................. 26  
3.2 UAV Development ........................... 29  
3.2.1 First Design ........................... 29  
3.2.2 Intermediate Designs ..................... 39  
3.2.3 Final Design ........................... 42  
3.3 Sensing Mechanism ......................... 46  
3.4 Localization ............................... 47  
3.4.1 Pozzyx Set Up ........................... 47  
3.4.2 Pozzyx Positioning ....................... 47  
3.4.3 Improving Pozzyx Accuracy .................. 48  
3.4.4 Implementing Pozzyx on Multiple Vehicles .................................. 49  
3.5 Implementing the IPAS Algorithm .............. 50  
3.5.1 Communications .......................... 50  
3.5.2 Sensor Relocation Algorithm .............. 50  
3.5.3 Actor Vehicle ........................... 51
4 Results and Discussion

4.1 Field Projection
   4.1.1 Final Projection Set Up
   4.1.2 Sensor Readings

4.2 Pozyx Precision and Accuracy
   4.2.1 Position Precision
   4.2.2 Position Accuracy
   4.2.3 Effect of Anchor Measurement Errors

4.3 IPAS Algorithm
   4.3.1 Sensor Relocation Algorithm
   4.3.2 IPAS Execution

5 Localization Using Wireless Signals

5.1 Introduction
5.2 Summary of the Problem
5.3 Solutions to the Problem
   5.3.1 Mapping
   5.3.2 Analytical
   5.3.3 Combination of Analytical and Iterative
5.4 Scalability into Three Dimensions
List of Figures

1.1 A diagram from the "Interactive Planning and Sensing in Uncertain Environments with Task-Driven Sensor Placement" paper showing three iterations of the IPAS algorithm on a particular Gaussian threat field. .............................................. 9
1.2 Testbed architecture by Valenti et al. (2007). .................................................. 11
2.1 Design rationale for each subsystem of the IPAS simulation. ............................... 13
2.2 An example of a threat field projection using different colors to represent threat intensity. 14
2.3 A diagram of the projection setup design, with two projectors on each side of the threat field, being fed a split image from a central network folder. ......................... 15
2.4 A simple sketch of the initial projector mount design, to be constructed of 3-D printed nylon supports and wooden dowels. ................................................................. 16
2.5 The Crazepony propeller selected for use on UAV design. .................................. 17
2.6 The Crazepony Martian II RX220 frame selected for use on UAV design. .............. 17
2.7 The Makefire 4-in-1 electronic speed controller selected for use on UAV design. .... 18
2.8 The Spektrum DSMX receiver selected for use on UAV design. ............................ 19
2.9 The Polhemus G4 localization unit sensor, hub, and anchor .................................. 21
2.10 The Pozyx localization unit anchor and tag ....................................................... 22
3.1 Projection from Alden Hall’s balcony using two projectors. ................................. 25
3.2 2-D and 3-D designs for the projector mounts, as seen from the bottom ............... 26
3.3 Example of a randomly generated threat field. .................................................. 27
3.5 UAV cutaway view showing relative component positions .................................. 29
3.6 Render of the UAV leg, with the screw-in point on top ..................................... 30
3.7 Render of the battery holder .............................................................................. 30
3.8 Render of the 9mm standoff .............................................................................. 31
3.9 Render of the X frame bottom support .............................................................. 31
3.10 Render of the X frame upper support and PixRacer enclosure ......................... 32
3.11 Render of the X frame Raspberry Pi 3 mount and upper frame support lid ........ 32
3.12 Render of the X frame Pozyx holder top and bottom sections ............................ 33
3.13 Kyosho Z8006 Zeal Vibration Absorption Sheet and the PixRacer flight controller mount. ............................................................................................................ 33
3.14 Without gel, dampening vibration issues seen by high VIBE. Vibe measures and VIBE. Clip values being nonzero. ......................................................... 34
3.15 With gel, Vibration Measurements and VIBE: Clip being zero ...................... 34
3.16 Single Motor Draw ....................................................................................... 36
3.17 All motor Current Draw without the extrapolated data of 80%, 90%, 100% throttle 36
3.18 Current draw minus the base. ....................................................................... 37
3.19 The Z-direction and Y-X-direction left and right CG offsets ......................... 38
3.20 X-Y offset images superimposed over each other ........................................... 38
3.21 Propeller thrust test configuration without the addition of a duct. ................. 39
3.22 Propeller thrust test configuration with the addition of a duct. ....................... 39
3.23 Propeller duct cross-section dimensions. ...................................................... 40
3.24 Current draw of the ducted (blue) and free (orange) propellers compared against measured thrust. .......................................................................................... 40
3.25 MB1043 HRLV-MaxSonar-EZ4 mounted on the UAV with a smoothing capacitor soldered on to it. ................................................................. 41
3.26 DROK LM2596 Buck Converter located on the rear of the drone. .................. 41
3.27 The final assembled UAV design with all components. ............................... 42
3.28 Images of the drone frame during assembly, before, during and after epoxying the Raspberry Pi mount to the arms. .......................................... 43
3.29 3D printed mount for the flight controller to sit. ....................................... 44
3.30 ESC mount and lower frame hardpoint. The smaller risers in the middle are where the ESC mounts. The outer larger risers are where the Lower Frame connects. The small holes are the screw in points to connect it with the main body. ............... 44
3.31 The lower frame both before it was melted together, and afterwards holding all of the components. ........................................................... 45
3.32 Dremel set up for the fabrication of the frame arms. Currently shown is the grinding to 5 mm thickness. ..................................................... 46
3.33 Layout of the Pozyx anchors ............................................................. 47
3.34 Wood mount for Pozyx anchors .......................................................... 48

4.1 Final threat field set up, utilizing four projectors mounted on stands atop tables, and the white paper background for enhanced image and data quality. .............. 52
4.2 Final mounting bracket configuration, with 3-D printed components and wooden dowels. 53
4.3 Projected color inconsistency using two different projectors. ........................... 53
4.4 Projected threat field representation using a grayscale ................................. 53
4.6 Pozyx anchor layout for accuracy test .................................................... 55
4.7 Pozyx X-coordinate localization measurements in mm ............................... 56
4.8 Pozyx Y-coordinate localization measurements in mm ............................... 56
4.9 Pozyx Z-coordinate localization measurements in mm ............................... 56
4.10 Pozyx localization error due to error in anchor placement ......................... 57
4.11 Greedy relocation algorithm paths ...................................................... 58
4.12 Alternate relocation algorithm paths ..................................................... 58

5.1 Example map of signal strength at given points. ....................................... 61
5.2 Trilateration with six unique intersections. ............................................ 62
5.3 Perimeter vs scale factor. ........................................................................ 63
5.4 Example of three positions and corresponding signal strengths. .................. 63
5.5 Example of three positions where there are two solutions. .......................... 64
5.6 Sample results 1. .............................................................................. 65
5.7 Sample results 2. .............................................................................. 66
Chapter 1

Background

In the recent years, civil and military applications for autonomous vehicles have expanded significantly, including data collection for research purposes as well as more critical missions such as surveillance and search-and-rescue operations. Autonomous ground vehicles (UGVs) have been developed for a variety of applications, from nuclear site maintenance and coal mining to space exploration. Similarly, unmanned aerial vehicles (UAVs) have proven particularly useful for remote field sensing in a variety of configurations. These configurations range from multiple UAVs as a cooperative perception system for detection of forest fires [1] to a single micro-UAV for assessing plant health in agronomic research [2]. Both aerial and terrestrial autonomy are therefore crucial for the development of critical missions in a wide range of applications.

The Interactive Planning and Sensing (IPAS) algorithm has potential applications in these type of time-critical operations, and its unique approach and effectiveness is the primary focus of this project. Unlike many other algorithmic approaches, IPAS is a task-driven method of information gathering that allows a mutual interaction between a sensor network and a path-planning algorithm, allowing an iterative process of sensor placement and motion planning to gather data more efficiently. Using this method, the sensors can be repositioned to focus on the areas of interest, providing more concise data and a decisive path using a limited number of sensors and iterations.

IPAS could prove to be advantageous in situations such as search-and-rescue or emergency vehicle navigation. In a wildfire rescue scenario for example, satellite imagery would provide cursory data about threat locations, and the algorithm would then refine the satellite data to increase the confidence of the data in the regions that matter most - around survivors and their evacuation route. For emergency vehicle navigation, historical traffic data would be used to establish high traffic areas as the basis of threat sources, followed by the use of IPAS to refine the route to the destination by delivering more accurate data of road conditions along the path. All these types of situations revolve around time-sensitive information gathering; the IPAS algorithm provides the task-driven information gathering required. This project implements and further develops the IPAS algorithm, both proving its effectiveness and providing a guide for how others can implement the algorithm to solve their own task-driven information gathering needs.

1.1 Project Description

This project describes the design and construction of a system involving the use of autonomous vehicles to implement the IPAS algorithm. Specifically, the system consists of a network of sensing UAVs gathering data to estimate an optimal path through a threat field for use by a ground vehicle (the "actor" vehicle). The intention is to produce a full physical demonstration of the IPAS simulation that incorporates both air and ground vehicles working under commanded actions given by the algorithm. To achieve complete vehicle autonomy, two components are crucial: situational awareness of the vehicles environment, and motion planning and control. These components are
incorporated by the observed data at a known location by means of the UAV sensing network and a motion-tracking localization system, and the feeding of this data into the IPAS algorithm for path coordinates. The system then coordinates a collision-free execution of the maneuvers given from the algorithm until the actor vehicle has traversed the optimal path to its destination.

This project is a direct physical application, demonstration, and further development of the IPAS algorithm described in the paper "Interactive Planning and Sensing in Uncertain Environments with Task-Driven Sensor Placement" [3]. Drawing on research in sensor management for the estimation of distributed conditions, motion planning for mobile vehicles, and coordination strategies for mobile sensors, the IPAS algorithm seeks to find the safest path through a threat field at the least expense.

1.2 IPAS Algorithm

The aspiration of the IPAS algorithm is to intelligently gather data for path planning using a small number relocatable sensors within a threat field. In essence, it is a method for gathering information in a task-driven manner rather than a knowledge-driven manner by allowing a bidirectional communication between the sensing network and path planning algorithm. IPAS attempts to prioritize data collection based on factors that make certain points more likely to have a greater impact on the path. Part of the solution to achieving this was to model the threat field as a sum of Gaussian distributions rather than an array of grid points. This allows for mathematical models to aid in the relocation of sensors, and for the number of sensors required to accomplish the task to be reduced. Modeling the field in this way permits a cross-correlation to be performed between the generated path and the Gaussian model, resulting in a ranked set of Gaussian peaks that are likely to have the most impact on the path. Placing sensors near these highly correlated components gives more relevant data to the task at hand. Figure 1.1 comes directly from the original IPAS paper [3], and illustrates the utilization of the algorithm on a random threat field. The original path coordinates and sensor locations are indicated, and the repositioned sensors and reoptimized coordinates can be seen for each iteration, observably closer to areas higher interest each time.

The algorithm needed several adjustments and adaptations to translate from a computer simulation to a full implementation. The sensor locations were given only as a batch of coordinates in each iteration, and did not consider each sensor to be a unique object to be physically relocated as a full implementation would. For this reason, a practical method of repositioning sensors had to be created that would efficiently relocate them while avoiding interference and collisions between them. The physical implementation also includes the actor’s movements through the threat field. The algorithm again had no support for this, as the path that it outputs is generated based on a fixed start and end location for the actor without accounting for its movement between iterations. Considering that this project’s objective is to demonstrate the algorithm utilizing autonomous vehicles, additional programming was required to compensate for these missing elements.
Figure 1.1: A diagram from the "Interactive Planning and Sensing in Uncertain Environments with Task-Driven Sensor Placement" paper showing three iterations of the IPAS algorithm on a particular Gaussian threat field.
1.3 Related Work

Over the past several years there have been many projects aimed at developing the most effective methods for utilizing UAV sensing technology in practical applications. This generally includes optimizing sensor recognition, path generation, and coordinated trajectory and motion planning of multiple UAVs. The most notable related work is an application of the Measurable Augmented Reality for Prototyping Cyber-Physical Systems (MAR-CPS) from the Massachusetts Institute of Technology. The MAR-CPS project parallels the IPAS project in that it seeks to provide a system that can be used by autonomous vehicles to accomplish complex coordinated planning and sensing tasks. Specifically, the goal of both projects include using a UAV and image recognition to guide a ground based actor vehicle through an obstacle course by scanning ahead for threats to be avoided, and then informing the actor vehicle of the most favorable path [4]. The IPAS project has a remarkably similar set up as MIT’s project, closely mimicking their physical field construction and use of both a UAV and UGV, but they differ in the intended use of multiple UAV coordination and simultaneous planning, sensing, and execution of the IPAS algorithm.

UAVs have been gradually adopted as sensing systems for a wide range of applications from agricultural studies to terrestrial moving target observation. This is due to both the perspective they can offer and the higher range they can encompass compared to ground sensing systems. A wide variety of sensing payload can be incorporated onto a UAV such as camera, thermal, laser, or spectral sensors to carry out different activities like mapping, modeling or monitoring. Moreover, with further development, multiple UAVs can be coordinated and perform sensing tasks cooperatively, thus encompassing larger fields, reducing task time, or providing greater accuracy through redundancy.

In 2017, Adao et al. proposed the use of hyperspectral sensors on-board UAVs to conduct agricultural and forest research or commercial activities. They highlight the benefits of this type of sensor over RGB imagery as primarily the higher precision, and wider spectral range for material and organism profiling. The authors also put an emphasis on the benefits of UAV remote sensing in terms of cost, availability, and logistics when compared to manned aircraft missions. Additionally, they studied the variety of sensors within this hyperspectral type, and lay out the necessary pre and post-flight operations, and calibration this sensing system needs to provide accurate data. Finally, this paper addresses the abstraction of the low-level complex mathematical processing hyperspectral sensing entails by presenting a variety of toolboxes that allow the front end user to easily acquire the resulting data from the sensors [5].

A similar application of UAV sensing was proposed by Yang et al. in the same year. The authors implement a system that consists of a Plantower PM2.5 laser sensor on-board a DJI Phantom 3 UAV to monitor and map fine-grained air quality index (AQI) in real-time. They study a novel particle dispersion model called Gaussian Plume model embedded in Neural Networks to predict the AQI of unmeasured areas. Secondly, they develop an adaptive monitoring algorithm to sense AQI efficiently by guiding UAVs through an optimized trajectory based on the particle dispersion model. This algorithm produces a great impact on power consumption reduction and real-time mapping of the AQI of an area [6].

Another concept within the field of UAV sensing, is the incorporation of multiple UAVs with the purpose of reducing task time, increasing range or providing another type of optimization. Avellar et al. solve an optimization problem related to the number of UAVs needed to cover and remotely sense an area in minimum time. They pose the problem of first, identifying the number of UAVs needed to minimize the time taken to cover and sense via cameras an area modeled as a convex polygon, and second, defining the routing scheme for each UAV such that their paths help complete the mission in minimum time. The main contribution of this paper consists of the analysis of real-world applications of this problem, through physical experiments, instead of mere simulations. The authors concluded that the optimal number of UAVs is a function of not only, the area to cover, but also of flight time and setup time, as well, as number of human operators available. It is important to note, that this paper did not find a solution for the case when the number of UAVs and battery life is small compared to the size of the area to cover. This is a problem, the IPAS algorithm proposed
in the present paper, attempts to solve by implementing an optimal sensor placement strategy [7].

An important component of multi-UAV sensing is flight coordination to avoid collisions. The study presented by Avellar et al. failed to take into account the possibility of paths intersection. However, notable research has been done in this area such as the work presented by Sinha et al., who propose the use of a group of UAVs to track the dispersion of a contaminant cloud in urban regions. They develop an algorithm to track, and estimate the boundaries of the cloud, and its shape, as well as to predict the movement of the cloud. This UAV swarm carries on-board sensors to track nuclear, chemical, and biological contaminants. The algorithm developed covers the path planning and UAV path collision problem. To address the latter problem, the cloud is modeled and divided into \( n \) sectors, each having two vertices. Each UAV is assigned to track the area encompassed by a pair of vertices, and thus, avoidance is ensured while each UAV only senses the cloud within the sector assigned [8].

More controlled environments or testbeds have also been developed to study and improve UAV flight tests and multi-UAV coordination for a variety of missions. One example is the work presented by Valenti et al., who present an indoor multi-vehicle flight testbed designed to study, under a controlled environment, different parameters of long duration missions. Their main goal was to incorporate health management considerations into mission planning, such as refueling and repair processes that may need to take place during a certain mission. The testbed architecture proposed by Valenti et al. is shown in Figure 1.2 and their main contribution consists on the incorporation of a health information component which provide feedback to the system regarding each actor performance and modify the mission accordingly [9].

![Figure 1.2: Testbed architecture by Valenti et al. (2007).](image)

A similar controlled environment contribution is given by the MAR-CPS project mentioned at the beginning of this section [4]. One common feature of these two testbeds is the indoor localization system used. Both of these projects used a Vicon Position System which uses motion capture to generate the current location of vehicles or moving components. While this technique is very accurate in generating indoor positioning, it is worth investigating alternatives for indoor localization systems mainly due to budget constraints.

An alternative to motion capture systems is the use of wireless signals in a variety of modalities, such as Received Signal Strength Indicator (RSSI) or Channel State Information (CSI). Yang et al., in their article, highlight key differences between these two position tracking modalities. RSSI uses the attenuation of radio signals as a mean for positioning. However it is mainly accurate at a room level and its performance decreases dramatically due to multipath fading. Thus, the need for a finer-grained wireless channel measurement. Channel State Information describes how a signal propagates from transmitter to receiver in more detail and hence it has shown to provide greater accuracy compared to RSSI [10]. According to the survey presented by Yang et al., CSI-localization can address the multipath issue encountered with RSSI ranging while decreasing error to around 2 m [11].

Wireless indoor localization techniques extend to the use of acoustic signals, infrared, RFID,
bluetooth, cellular, WiFi, UWB and others [10]. For an indoor UAV testbed, a high accuracy (around centimeter-level accuracy) is desired. According to Zwirello et al., short ultrawideband pulses are resistant against multipath effects unlike RSSI. Also, UWB systems are required to transmit at very low power, hence the short-range application. The authors present an indoor localization demonstration using UWB, in conjunction with the time difference of arrival (TDoA) method which requires information about time propagation from tracked devices to access points. To calculate position of tracked devices, a series of algorithms for reducing the error in data were analyzed and the best option was selected in terms of accuracy and computation time [12].
Chapter 2

System Design

Implementing the Interactive Planning and Sensing algorithm into a physical system requires the design and construction of the following components: a visual rendition of the threat field, the hardware to be used for mobile sensing, the hardware to be used for path execution, the localization system for coordinated guidance, and the accompanying software and method of communication between the devices. The design rationale for all of the major decisions made during this project can be seen below.

<table>
<thead>
<tr>
<th>Subsystem Name</th>
<th>Design Requirements</th>
<th>Options considered</th>
<th>Final Design</th>
</tr>
</thead>
</table>
| Thread Field Representation | - Easy visualization  
- Represent a field that can be sensed by accessible sensors | - Projectors  
- Floor mat  
- Sound frequencies  
- Infrared light | - 4 to 4 projectors  
- Mounted on speaker stands with a custom made 3D printed frame  
- Projection is onto the floor covered by long sheets of white bond paper for clearer visualization |
| Localization System      | - Indoor centimeter accuracy  
- Easy to integrate software for localization | - Motion Capture System  
- Polhemus 04  
- Pozzy | - Pozzy system  
- 4 actors placed on custom made wood stands  
- Pozzy tags on UAVs and ground vehicles |
| Server and communications | - Easy to setup and to work with  
- Scalable and asynchronous  
- High speed and reliable | - WiFi or bluetooth.  
- HTTP or TCP sockets. | - Communication server works over WiFi due to ease of use with multiple devices.  
- Used both HTTP and TCP sockets. HTTP was used for data transfer and large operations and sockets were used for time sensitive, low-level data |
| Image Sensing System     | - Low-cost sensors  
- Light-weight, easy-to-mount sensors on UAVs and ground vehicles  
- Simple integration with the rest of the system via the communications system | - Thermal lasers  
- Spectral sensors  
- Cameras | - PI Camera mounted on UAVs and ground vehicles  
- Configured to take frames and post them to the server |
| UAV design and implementation | - Flight time between 10-20 minutes  
- Hold a pose in the vertical position  
- Fly a Raspberry Pi 3 B on it  
- Be less than $300  
- Be less intimidatingly large | - Ready to fly UAVs (from DJI)  
- Almost ready to fly UAVs  
- 180mm custom quadrotor  
- 220mm custom quadrotor | - UAV with individually selected parts  
- Custom made frame  
- 5-inch propellers  
- 40g of weight  
- 1800mah battery |
| Ground vehicle design and implementation | - Navigation to waypoints  
- Field Sensing  
- Integration with server through some protocol   
- Drone | - Four Wheel Skid Steering  
- Two Wheel + Castor Skid Steering  
- Camera viewing local projection surface  
- Camera on boom arm | - Two Wheel + Castor frame with boom arm camera |
| IPAS algorithm            | - Run on raspberry pi  
- Computationally light and adequate language for RAM  
- Easy integration with other subsystems such as communications and image sensing | - Headless version of MATLAB, no GUI  
- Implementation in Python or C | - Fully prototyped in Python  
- Heavily dependent on Numpy and Scipy libraries  
- Works on any field size or resolution |

Figure 2.1: Design rationale for each subsystem of the IPAS simulation.

2.1 Field Projection

A primary aspect of a physical demonstration of the IPAS algorithm includes setting up a threat field that can be measured by sensors. This had to be something that would consistently provide precise data, and preferably contain a visual element so the threats could be detected by direct observation. While there are numerous methods of providing a field capable of the intended purpose, such as using infrared light or various sound frequencies with the associated sensors, it was concluded that the best representation would allow a visual reference of the threat field. This way, any observer of
the demonstration can distinctly recognize where the peaks of the threat are, and how the sensors reposition themselves relative the greatest threat intensities and the optimal path.

To accomplish this visual display, options such as a full-scale printed floor mat or height-varying platforms could have been considered, but it seemed most logical to take inspiration from MIT’s MAR-CPS set up of a ceiling-mounted projection system [4]. Projectors were ultimately chosen due to their simplicity, ability to render dynamic fields, and previously validated success.

### 2.1.1 Threat Field Representation

To satisfy the sensing capability and visual requirements for the threat field, the simplest design would be a flat projection of the field on the ground using different colors or hue intensity to represent the magnitude of the threat. Red, for example, could suggest an area of high threat whereas blue would suggest an area of low threat and green may suggest something inbetween, as exemplified in Figure 2.2. Alternatively, a monochrome display could be used where white areas suggest a low threat, and one color would increase in intensity in the areas where the threat becomes more severe. Considering that for a colorized visual display the most fitting sensor mechanism would likely be a camera, it was apparent that a camera can decipher a predominately red area versus a blue area with much more precision than a light blue area versus a dark blue area. Therefore it seemed appropriate to use the method of multiple colors for various threat intensities, with complementary colors chosen as opposite ends of the intensity spectrum for greatest accuracy.

![Figure 2.2: An example of a threat field projection using different colors to represent threat intensity.](image)

### 2.1.2 Projectors

When selecting the hardware required for this type of projection, the first consideration made was the size of the displayed field. To provide enough space for multiple automated vehicles to travel without collisions, the initial objective for the fully developed system was estimated to be approximately 10x10 m$^2$. There are very limited projectors capable of displaying an image of this size clearly, and those that can tend to be relatively expensive. Most projectors have a maximum display size around 300 in, but to obtain a 10x10 m$^2$ field the display size would have to reach close to 400 in. To accommodate for this, the decision to use multiple projectors threaded together to create one cohesive image was made. This would also assure the luminosity of the image is not depreciated as a result of the large throw distance required to cast an image of that size with a single projector.
By utilizing four projectors with a display size of about 200 in each, the 10x10 m$^2$ image can be achieved with clear resolution.

With more leniency in the projector selection this way, it was a simple decision to use the projectors provided from Worcester Polytechnic Institute for no cost, specifically being NEC M series models. The specifications of these projectors match the criteria, as well as save on budget costs. Taking a simple approach to implementing multiple projectors, a short Python script was devised exploiting the Python Imaging Library (PIL) to divide and distribute a single image, shared on the network folder, to the four respective projectors.

2.1.3 Mounts and Set Up

The most critical considerations when determining the best positions for the projectors were achieving the target size of the field, and the interference of shadows the hovering UAVs may cast on the field below them. While the field size versus readily available projector display sizes was discussed in the previous section, the UAV shadows poses an additional issue. A great deal of time was dedicated to the consideration of short throw projectors, which would allow the projector to be mounted nearly on the ground below the hovering UAVs, eliminating any shadow complications. These projectors, however, also prove to be excessive for the project budget, so high-mounted stands holding long throw projectors were chosen. To assure the UAV shadows have as little impact on the field projection as possible, the projectors are mounted at an angle around the field perimeter as opposed to directly above the field projecting down, as depicted in Figure 2.3. While this reduces the maximum potential size of the field, having four projectors with slightly smaller displays suffices for the 10x10 m$^2$ goal if mounted high enough, and significantly reduces the influence of shadows. Each projector requires a set keystone correction to adjust for the angle, providing a evenly rectangular field. In order to securely mount the projectors at the proper height for a large display, found to be approximately 10 m for a 10 m tall image, speaker stands were decided upon because they are designed to suspend a large load at a variety of heights for an extended amount of time. The mounts on the stands, however, do not match those required for a projector, so a bracket device was designed and 3-D printed to fit the two components together. The bottom of the projector has four inserts for MAX M4x8 screws, while the stands have a standard 1-3/8 in shaft for mounting speakers. The following diagram was drafted as a basis for the required mounting bracket. The mount is comprised of small 3-D printed supports with the proper sized threaded holes for screws, connected to wooden dowels which hold a centerpiece capable of fitting properly on the stand shaft. This centerpiece holds the projector at a fixed angle, as opposed to perfectly perpendicular to the shaft, so that the images are consistently projected downward at the proper inclination, and adjusted with the proper keystone angle.

![Figure 2.3: A diagram of the projection setup design, with two projectors on each side of the threat field, being fed a split image from a central network folder.](image-url)
2.2 Field Sensing

2.2.1 UAV Design

A UAV typically consists of eight main components that create a device that flies: the frame, battery, motors, propellers, electronic speed controller, power distribution board, flight controller, and receiver. Designing a UAV is a highly iterative process as certain component selections change how other components will need to be sized. First, the requirements the UAV would need to satisfy were defined. The fundamental constraints established include: the flight time must be at least 8 minutes, the payload must include a microcomputer and localization hub (later determined to be a Raspberry Pi 3 and Pozyx tag, explained in Sections 2.3 and 2.4), and the overall size must be minimized to comply with field size constraints.

Propeller

Propellers for UAVs have three measurements: tip to tip diameter, pitch, and blade count. The first factor to decide on is the tip to tip diameter because it limits the efficiency and feasible payload of the UAV. After looking into 3, 4, and 5 in propellers, it was found that a larger propeller has a greater surface area, which allows it to push more air while spinning at a slower rate. The larger surface area provides a greater lifting capacity, and the slower spin rate causes less drag, reducing wasted power. Considering the relatively large payload required, the options were narrowed down to 5 in propellers to avoid issues with carrying capacity and flight time. The pitch, which is the distance a propeller moves forward after completing one full rotation, is almost exclusively found to be 3 in for a 5 in propeller. There are also 4.5 in pitch propellers, but a higher pitch can lead to a vortex ring state and potential loss of control. Lastly, the blade count has a major impact on the lifting force, where more blades increase maximum lift, but decrease the efficiency of thrust [19]. With limited power and an already determined 5 in propeller diameter to assure proper lift conditions, a lower number of blades may be optimal for this application to increase power efficiency.
One last decision for propellers is the material, of which there are two readily available to choose from being plastic or carbon fiber. Plastic propellers are cheaper but are not as rigid as carbon fiber propellers. The rigidity is significant because a more flexible propeller will wobble more during flight and produce uneven lift during its rotation, leading to undesired vibrations. While this effect is minor, it is best to avoid for a small increment in price.

Ultimately, considering all of the constraints and propeller characteristics, it was decided the ideal propeller would have a tip to tip diameter of 5 in, a pitch of 3 in, two blades, and be made of carbon fiber. These decisions led to choosing the carbon fiber 5030x2 propeller from Crazepony.

![Figure 2.5: The Crazepony propeller selected for use on UAV design.](image)

**Frame**

Having established the propeller dimensions, a frame size can be selected. Frames are measured by the distance the motors are diagonally apart from each other. To carry 5 in propellers, the frame must be at least 185 mm; however, this would be extremely tight and would not leave enough space to fit the rest of the payload including the Raspberry Pi 3 and Pozyx tag onto the UAV. Therefore, a 220 mm frame was selected because it leaves sufficient space in the center for all the components, and does not have the additional weight of a 250 mm frame. Unsurprisingly, almost every UAV frame on the market is composed of carbon fiber because it provides the best strength and durability for its weight [19].

Still, aside from this frame, the UAV arms have to be selected; UAV arms are separate from the UAV core so that broken arms can be substituted without replacing the entire frame. This modularity also allows for custom 3-D printed central sections, which in this case is necessary so that the Raspberry Pi 3, Pozyx tag and chosen flight controller can all fit within the UAV footprint.

With the UAV arms to be designed and fabricated later, the core frame selected was the Crazepony Martian II RX220, with a weight ranging from 89-137 g depending on what mounting hardware is required.

![Figure 2.6: The Crazepony Martian II RX220 frame selected for use on UAV design.](image)
Battery

The propeller selection also influenced the choice of battery cell count. Li-Po batteries are composed of individual cells that normally measure a nominal 3.7 V. However, a newer Li-Po chemistry is becoming commercially available that can reach slightly higher voltages: 4.35 V instead of 4.2 V. The higher voltage batteries are more expensive but are 12% lighter for the same energy capacity. Another advantage of high voltage Li-Po is voltage stability. Under high current draw situations, a higher voltage Li-Po experiences less voltage drop, creating a more consistent motor response. Battery capacities from 1300 mAh to 2000 mAh were examined, ensuring there would be enough current provided to power all the components [19].

In the end, a LiHV battery was picked over a Li-Po because of its greater stability and lower weight, ultimately saving 23 g of weight. A 1800 mAh capacity was chosen because it had the best energy density of the batteries available, and the greatest certified flight time over various payload weights (from 50-200 g). This project’s UAVs will utilize the Turnigy Bolt 1800 mAh 11.4 V 65 130 C LiHV pack, weighing 160 g.

Motor

The motor decision stems from both the propeller choice and the battery voltage. The motors most important characteristic is their KV rating, which is the speed at which the motor will rotate per volt under no load. For example, a motor with a KV rating of 1700 driven by 1 V will spin at 1700 rpm, or 17,000 rpm when driven by 10 V. The KV rating therefore determines the maximum propeller rpm because we know our battery has 11.4 V. Smaller propellers use a motor with a higher KV rating because they must spin faster to generate lift. The larger the propeller, the lower the KV rating the motor should have. Another important characteristic of motors is power rating. This is mostly determined by the physical size of the motor, but magnetics and coil parameters are also taken into account. Lastly, the motors’ hover throttle, the ratio of lift to maximum lift, should be in the 45%-65% range [19]. Otherwise, the UAV will either be too responsive or not responsive enough and make it difficult to control. The propeller KV rule of thumb assumes a constant voltage across all scenarios, which is no longer the case because we are using higher voltage Li-Po. The higher voltage only means we need a motor with a slightly lower KV rating, but comparable power rating so that the UAV is still close to 50% hover throttle.

After many iterations, it was settled that the EMAX MT1806 motors rated at 2280KV were the best option. These motors were selected because they are available in the USA, are rated for the maximum power the Crazepony propellers can deliver, have a low series resistance, and result in a throttle position of 47% which is still within tolerance.

Electronic Speed Controller

Figure 2.7: The Makefire 4-in-1 electronic speed controller selected for use on UAV design.

The electronic speed controller (ESC) is a UAV’s motor controller, converting the flight controller’s commanded thrust signal into the signal required by the brushless motors. Individual ESCs are required when power requirements are larger; however, 4-in-1s are suitable for smaller UAVs
This 4-in-1 format should be adequate for this system’s intended use since the chosen motor and propeller pair can only draw up to 8.5 A. This format will also slightly conserve weight for comparable performance. Therefore, the controller chosen is a 4-in-1 12 A ESC made by Makerfire.

**Power Distribution Board**

The power distribution board (PDB) is not dependent on the rest of the components. It serves to take power from the battery and distribute it to all the UAV elements, such as the ESCs, flight controller, and receiver. Some PDBs also have a battery elimination circuit (BEC) that steps down and regulates the voltage going to the flight controller and receiver. A BEC allows the components to be driven from the primary UAV battery eliminating the need for a separate, lower voltage, control battery. To save weight, the Matek Micro PDB was selected which includes a 5V and 12V BEC.

**Flight Controller**

Today there are a wide variety of flight controllers available on the market, all designed for different UAV applications, and supporting different software and processing powers. The first flight controller attempted for use was a PixHawk 2.1, a powerful flight controller designed for autonomous UAVs. The downside of this flight controller is its relatively high weight and unusual shape. While it might be ideal for a larger UAV, because of size constraints the flight controller choice was switched to PixHawk’s Pixracer. The Pixracer is a much smaller and lighter unit, about 1/3 the weight (11 g) and about 1/4 the volume. It is the most recent PixHawk developed and utilizes a 32-bit F4 processor, the most popular of the stronger processors available.

**Receiver**

The choice of receivers were decided by what communication protocols they employ. Five protocols exist: pulse width modulation (PWM), pulse position modulation (PPM), pulse code modulation (PCM), serial bus (S.Bus), and digital system multiplexor (DSM2 and DSMX). PWM and PPM are both analog signals and require a wire for each channel that will be connected to the flight controller. PCM is similar to PPM, but is a digital protocol and is more robust. S.Bus is a fully digital signal and has the advantage that it can support up to 18 channels through a single cable. DSM2 and DSMX are digital like S.Bus, but are even more resilient to interference and don’t have channel limitations.

![Figure 2.8: The Spektrum DSMX receiver selected for use on UAV design.](image)

The DSM2 and DSMX both use Code Division Multiple Access (CDMA). CDMA spreads the communication frequency across multiple bands, with each transmitter-receiver pair using their own encoding. This makes it resilient to static and allows multiple transmitters to be communicating on the same channel while the receiver is still able to isolate its pair’s data. DSMX adds another safeguard onto DSM2, where instead of transmitting at a few randomly selected frequencies and only communicating on those frequencies, its frequency hops. DSMX changes communication frequency
thousands of times per second so that even if one channel is unusable, the communication blackout only occurs for a few milliseconds [19].

The need for reliable communication is crucial because of the large potential for interference from other UAVs, the actor vehicle, and the Raspberry Pis Wi-Fi. While both the DSM2 and DSMX can provide unique frequencies to prevent interference, the DSMX affords an extra layer of resilience. For reliable communication and minimized cabling, a Spektrum DSMX receiver was selected.

2.2.2 Sensing Mechanism

Pi Camera

Each UAV in the IPAS system requires an on-board sensor that can evaluate the threat field color distribution projected on the ground beneath them. As stated earlier, the sensors are appropriately cameras so that some basic image processing work can be done to gauge the threat intensity. Since each UAV will already be equipped with a Raspberry Pi 3, it was evident that the most reasonable choice in camera sensor is a Pi Camera to ensure integration simplicity and verified compatibility. The 8-megapixel Pi Camera v2 has been mounted on the bottom face of each UAV, aimed downward to capture snapshots the projected field below.

Image Processing

To analyze the data from the Pi cameras, frames could either be processed on board the UAVs and returned as a numerical result to the path planner, or the path planner can process the frames itself. Either function can be implemented using Python with picamera, PIL, and NumPy. The frames are evaluated based on RGB color saturation, with one color being established as areas of high threat and the complementary color being areas of low threat. Since the received frames likely portray a wide image of the field with a potentially large variation in color distribution, the image processing focuses solely on a small section of pixels in the center of the image and averages their RGB values. This way, only the pixels directly below the sensor vehicle are averaged and interpreted as having a particular threat intensity for that position. This allows for more precise point data, which as of now is the only interpretation of threat field data the IPAS algorithm is able to handle. In future development when the threat field is simulated as a dynamic time-varying field, the algorithm would preferably be able to take in the entire image gradient from a frame and use predictive and task-driven methods to determine the optimal path.

2.3 Localization

A crucial step for achieving complete autonomy of the IPAS system was selecting and invoking an accurate localization system. This system must be able to determine position and orientation of all the major components in the system. In this project, these components include the UAVs that serve as field sensors, and the ground vehicles that serve as both the actor and as additional field sensors. The basic requirement for the localization system selection was to find a method of accurately tracking 3-dimensional position in the order of centimeters, due to the size small of our threat field.

One of the first design decisions of this project was that the physical demonstration of the IPAS system would occur indoors due to strict UAV flying regulations, among other reasons. Therefore, in order to track devices within the system, two different Indoor Positioning Systems (IPS) were tested for accuracy and reliability. Although these are complicated and expensive systems, other positioning systems such as GPS are not suitable for indoor use, as their accuracy is in the order of meters, which is too large for the scope of the system. First, the Polhemus G4 positioning system was introduced to the system, and after some testing, its performance was compared to testing
results of the newly released Pozyx positioning system to determine the best option.

2.3.1 System Selection

Polhemus G4

The G4 is a six degree of freedom motion tracking system developed by Polhemus which can track both position and orientation wirelessly. This system utilizes Polhemus proprietary AC electromagnetic technology allowing tracking without line-of-sight requirements [13]. G4 components include:

- Sources: generate the magnetic field for sensor measurements.
- System Electronics Unit (Hub): device that computes the position and orientation of up to three sensors and wirelessly transmits this data to the destination server, weighing 114 g.
- Sensors: small cubes whose positions and orientations are measured in real time, weighing 9.1 g.

The major drawbacks of this system are the high payload, small source range, and the generation of proprietary data. First, in order to track position and orientation of a device, both a hub and a sensor are needed on-board the device to be tracked. This meant an extra payload of approximately 123 g for the UAVs, which would significantly increase their price and size. Second, each source generates a circular field of an approximate radius of 3 m. The only four sources at disposal for the project would therefore be insufficient for generating the desired threat field size, and the purchase of additional sources and sensors signified an elevated cost. Finally, the real-time position and orientation data generated by the G4 product was produced in such a way that feeding it into the algorithm and sending it to the corresponding system components was very challenging due to proprietary issues. For these reasons, other options were analyzed for consideration of the localization system.

![Figure 2.9: The Polhemus G4 localization unit sensor, hub, and anchor](image)

Pozyx

Pozyx is an indoor localization system that utilizes ultra-wideband technology to wirelessly track motion of devices with centimeter accuracy. The Pozyx components include anchors which represent the sources that delimit the tracking field, and tags which are to be positioned on the devices to track. Pozyx tags come equipped with accelerometers, gyroscopes and magnetometers, which altogether compensate for each others flaws and provide accurate measurements of position and orientation within a 10 cm accuracy [14].

The Pozyx system outperformed the G4 system in every design parameter considered. First, Pozyx tags weigh approximately 12 g while the hub and sensor of the G4 system weigh around 123 g.
This meant a significant UAV payload reduction and hence a reduction in cost and size of the UAVs. Second, the maximum range of the ultra-wideband signals are 100 m in clear line of sight, while the G4 system could only achieve a circular coverage of a 3 m radius per source. Finally, a significant aspect in the localization system decision was the possibility of smoothly exporting data from the product to the other components of the system. Since the Pozyx system is an open-source product easily compatible with Raspberry Pi boards, and for all the other reasons aforementioned, the Pozyx system reigned far superior to the G4 system, and ultimately became the chosen localization system for this project demonstration.

![Figure 2.10: The Pozyx localization unit anchor and tag](image)

### 2.3.2 Mounts and Set Up

The ideal positioning for the Pozyx anchors according to the developers is to have each mounted at various heights around the perimeter of the threat field [14]. Since there are already four stands intended to be around the field holding the projectors, it is convenient to double these stands as the anchor mounts as well. This will allow variations in the height of the anchors along the vertical stands, as well as reduce expending more money and time to setup. Since the Pozyx anchors are light-weight and fitted with several holes for various mounting options, the anchors should be able to be zip-tied or velcroed at the established heights to the speaker stands.

### 2.4 Path Planning, Computations, and Communications

In order to easily network all sensors with the actor, each vehicle is equipped with a Raspberry Pi 3 connected to the same isolated Wi-Fi network.

#### 2.4.1 Raspberry Pi 3

Each Raspberry Pi 3 communicates with the vehicle’s on board systems, such as the PixRacer for the UAVs, the Pi Camera on all sensors, and the Pozyx tag on all vehicles. The data and power connections are made over USB for the PixRacer and Pozyx, whereas the Pi Camera has a dedicated camera port. On all vehicles, a background process is constantly running a Kalman filter over the localization data received from the Pozyx system to maintain accurate positioning data for when it is requested.

**The Actor**

The actor vehicle is a ground based robot commanded by a Raspberry Pi 3b, which communicates with the Command Unit by using a Unix socket. All communication is asynchronous to prevent downtime while waiting for responses. The actor vehicle is commanded to navigate to given waypoints by the Command Unit, and executes these commands by using a PID loop to control angular velocity, and by setting its linear velocity to be proportional to the distance to the target waypoint.
When the actor vehicle approaches the waypoint within the margin of error, the actor vehicle brakes, and waits for the next waypoint command.

\[ \omega_n = \omega_{n-1} + K_P(\Delta \theta_n) + K_D(\frac{\Delta \theta_n - \Delta \theta_{n-1}}{t_n - t_{n-1}}) + K_I(\sum_{i=0}^{n} \Delta \theta_i) \]  

(2.1)

\[ V = K_V || \delta X || \]  

(2.2)

The motor control function takes these values and converts them to angular velocities to write to the left and right wheels. The drive controller first calculates the angular velocities required for linear drive at the given velocity. Next, it calculates the required difference in wheel angular velocities to achieve the desired turning angular velocity of the vehicle, then adds half this value to one wheel and subtracts half from the other. Lastly, due to limits on the wheel angular velocities, it scales each value in proportion to the other such that the greater of the two angular velocities does not exceed the given limits.

\[ \omega_{lin} = \frac{V}{R_{wheel}} \]

\[ \Delta \omega_{turn} = \frac{\omega_{lin} \times \Delta \omega_{turn}}{R_{wheel}} \]

\[ \omega_{right} = \omega_{lin} + \frac{\Delta \omega_{turn}}{2} \]

\[ \omega_{left} = \omega_{lin} - \frac{\Delta \omega_{turn}}{2} \]  

(2.3)

The Command Unit

The Command Unit is the central controller for the demonstration IPAS system. This computer communicates to each sensor vehicle and the actor vehicle via HTTP.

The Sensors

On each sensor, an HTTP server using an Nginx -> Gunicorn -> Python -> Falcon stack is always running for remote control of the sensor. Functionality includes, but is not limited to requesting sensor data, requesting localization data, and repositioning the sensor. GET commands are used to retrieve information whereas POST commands are used to make changes to the sensor such as positioning changes.

2.4.2 IPAS Algorithm

The actor, in a separate process or sub-process, runs the IPAS algorithm which has been converted from MATLAB to Python so it can run on the Raspberry Pi. The algorithm outputs the optimal path in a format that can be interpreted by the actors navigation system, as well as new sensor locations to be relayed to and interpreted by the UAV navigation systems. This means the actor generates its own movement patterns and communicates the new set of sensor coordinates to the UAVs, creating a fully self-contained system. The algorithm then takes new formatted inputs from the sensors while maintaining memory of past inputs to iterate the process until the optimal path converges.

2.5 Path Execution

2.5.1 Actor Vehicle

The actor vehicle is where the majority of the processing occurs, and is the vehicle which traverses the threat field after the optimal path has been determined. In order to do so, the actor must be capable of localizing alongside the sensor vehicles so it can navigate itself to the points along the determined path. The vehicle is composed of its chassis, localization system, power distribution
system, battery, motor controllers, and processing system. The payload it will carry will consist of a wireless router for communication with the sensor vehicles.

**Chassis**

When debating on a design for the actor vehicle, it was clear that the vehicle would ideally be relatively simple and low in cost, since most of the budgeting is dedicated to the more intricate designs such as the UAVs and localization system. A UGV is typically less elaborate than a UAV, and there are plenty of kits available on the market for their basic construction. These kits vary in the number of wheels, number of motors, power requirements, and system accessories such as encoders. For the intended use in the IPAS system, a simple 4WD kit suffices, and the addition of encoders could aid in enhancing the positioning accuracy. Ultimately a DFRobot 4WD Arduino Compatible kit was chosen, and was modified to drive using two powered wheels and a rear caster wheel, eliminating the need to account for wheel slip in the drive code.

**Power Distribution, Battery, and Motor Controllers**

The two motors on the actor are controlled using a single two-channel motor controller capable of handling 3 amps at 5 volts. The motor controller uses a PWM signal to regulate motor speed, and an H bridge driven by two digital inputs to control motor direction. Power to the vehicle is supplied by a 2 cell 4000 mAh lithium polymer battery. Voltage to the Raspberry Pi, encoders, and motor controller is supplied by a step down voltage regulator, and voltage to the motors is supplied directly from the battery.

**Localization System and Processing**

The processing and localization systems on the actor are as previously discussed, utilizing a Raspberry Pi 3 and a Pozyx tag to keep consistency with the rest of the IPAS system. The Raspberry Pi has a Python adaptation of the IPAS algorithm, which is used to communicate position changes to the sensor vehicles, as well as direct the actor’s motor controllers to follow the outputted path. By utilizing the combination of the Pozyx system with data from the wheel encoders on the vehicle chassis, a more accurate estimate of the vehicle position may be acquired to ensure errorless navigation.
Chapter 3

System Development

3.1 Field Projection

3.1.1 Projection Set Up

In order to provide enough space to exhibit the IPAS system and adhere to flight regulations, it was decided to conduct the physical demonstration in Alden Hall at Worcester Polytechnic Institute. As previously stated in the System Design section, initially the projectors were intended to be located along the perimeter of the threat field at a height of approximately 10 m. After some preliminary tests, however, an alternate option was discovered to place the projectors on the balcony available in Alden Hall. This way, they were undoubtedly able to achieve sufficient height for the large scale projection sought, and still project at an angle to avoid interruption from shadows. Figure 3.1 shows a projection test done from Alden Hall’s balcony using only two projectors on the wooden floor.

![Figure 3.1: Projection from Alden Hall’s balcony using two projectors.](image)

Unfortunately, once projection from the balcony was further experimented with using all four projectors, it was found that the keystone correction and focus adjustments were not enough to compensate for the misalignment in images. A second balcony would be required to obtain a symmetric...
field, so it was concluded that the best plan of action would be to elevate the stands using tables on
the ground. Now, with the stands once again distributed evenly about the perimeter of the threat
field and at a height capable of obtaining a 6x7 m² image, the projection was solidified. Although
this does not meet the initial goal of a 10x10 m² field, it is still adequate for the intended purposes
of the system. To make the clarity of the image even more distinct, white paper was placed on the
ground to improve the camera sensors’ data quality.

In order to coordinate the four projectors to display one large image of the threat field evenly
and obtain a smooth transition along the edges of each projection, a Python script was developed
to split an image into four sub-images. This split function takes both the filename of the threat field
image saved in the network folder and the number of projectors used as inputs, so that the script
could be used for any number of projectors, independent of the image resolution. This was done by
applying the existing Python library Image Slicer, which splits an image into $n$ equally sized tiles.
These tiles or sub-images are then sent via HDMI to each projector.

3.1.2 Projector Mounts

The projectors’ mount design was influenced by the rudimentary sketch seen in Figure 2.4, and is
composed of 3-D printed supports made of Taulman Alloy 910, and 1/2 in diameter wooden dowels.
Each one of the projector’s bottom four screw holes has a 3-D printed piece which has a thru-hole to
insert the required M4 screws, and two side connectors to receive the dowels that link the supports.
This can more clearly be seen in Figure 3.2, showing both a rendered 2-D and 3-D representation
of the device. The dowels are friction fit into the vertex pieces but are glued into the left and right
side pieces. The design has a 5/4 in diameter female connector tube to fit on top of speaker mounts,
which originally was to be printed at a fixed angle. Since then, a new scheme was composed so
that the angle may be adjusted and affixed. With this adaptation, the centerpiece controls the
projection angle by rotating around the center dowel rod and is locked into place by screwing 1 in
wooden screws into the side holes. The axis of rotation is roughly below the center of gravity of the
projector when pointing horizontally, so once the angle of projection is finalized, the center dowels
length can be adjusted to shift the projectors center of gravity above the centerline of the speaker
stand for optimal stability.

![Figure 3.2: 2-D and 3-D designs for the projector mounts, as seen from the bottom](image)

3.1.3 Threat Field Representation

The threat field is assumed to be a collection of peaks represented by Gaussian functions, as explained
in the IPAS Algorithm section of the Background, where the threat level is linearly related to the
value of the function. As a means to simulate several randomized threat field images mirroring the
example shown in Figure 2.2, Python, NumPy, and matplotlib were used to generate images like the
one seen in Figure 3.3.

![Figure 3.3: Example of a randomly generated threat field.](image)

These images can then be stored on the network folder and divided for projection. Each threat was rendered using the general form found in Equation 3.1 where \( x_0 \) and \( y_0 \) are the randomly selected centers of the threats and \( A, \sigma_x, \) and \( \sigma_y \) are randomly generated parameters for the amplitude and size of each.

\[
f(x, y) = A e^{\left[-\left(\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2}\right)\right]}
\]  \hspace{1cm} (3.1)

Once the threat field is rendered and projected, the colors are picked up by the cameras on-board the sensing vehicles and translated back into threat values, which will be fed into the IPAS algorithm.

The colormap to represent the threat field was chosen from the options available within the matplotlib library in Python. An important component of the threat field representation is how visually understandable it is to the spectator of the simulation. For this reason, a perceptually sequential colormap was chosen to represent threats in the field. This means that the transition from color to color is smooth to the eye as illustrated in Figure 3.4a. Figures 3.4b and 3.4c show generated threat fields with the same number of threats, represented using miscellaneous colormaps (from matplotlib) named hsv and jet respectively, to show the difference between perceptually sequential and miscellaneous colormaps. The colormap chosen for this project’s threat field representation is named plasma and contains 256 different colors [15].
(a) Plasma colormap.

(b) HSV colormap.

(c) Jet colormap.
3.2 UAV Development

3.2.1 First Design

Component Positioning

Deciding where components such as the battery, flight controller, Raspberry Pi 3, Pozyx, and the ESC were the first decisions made during the UAV design. The battery had no specific requirements for mounting other than it being easily removable and secured. The flight controller, PixRacer, had to be mounted as close to the center of gravity as possible and fully protected on all sides. The F.C. should also be kept away from the ESC and battery to help reduce magnetic interference with the magnetometer during flight. The Raspberry Pi 3 did not have specific positioning requirements, but its large size required the UAV frame to be slightly expanded from the original 220mm cross section. The Pozyx tag being heavily reliant on strong signal strength had to be mounted on the external stack, either the bottom or the top, to maximize the signal strength it would receive from the anchor beacons. The ESC needed to be centrally located very close to the plane of the motors as it was a 4in1 format and having it offset would just over complicate matters.

The initial design layout had the Pozyx on top, then the RPi, PixRacer, Frame X, Esc, and on the bottom, the Battery. The battery was chosen to go on the bottom to help pull the CG lower because the RPi, and Pozyx tag would be rather high relative to the plane of the motors. Seated between the battery and the frame sat the ESC. The ESC and the battery were on the other side of the frame than the F.C. to slightly reduce the magnetic interference it sees. The separation also provided a convenient location magnetic shielding could be added later if it proved necessary. The flight controller was mounted above the frame on top of vibration dampening gel. The RPi was positioned above the flight controller to assist in protecting the F.C., and the RPi's mount served as a ceiling for it. The RPi's CG was offset from the axial line from the UAV so that the cross-section of the board was in the center of the quad. Unfortunately, the RPi's center of mass was not in the center of its board. The RPi's CG was not centered with the UAVs because the extent the board would need to be offset to shift the RPi's CG above the center line would cause negative aerodynamic effects on the propellers. Finally, the Pozyx tag was mounted above the RPi. The complete assembly of the first iteration of UAV design is shown in Figure 3.12 below.

![Figure 3.5: UAV cutaway view showing relative component positions](image-url)
Component Design

The individual components’ design was an iterative process focused on making slight tweaks to shift how the CG would center, to provide additional space to some components, and to improve manufacturability. Following is a breakdown of each part that is 3D printed and a short explanation for each design decision made at the time.

**UAV Legs**  The legs were curved to help absorb energy in a hard landing. It only has one screw in point because there is only one hole in the carbon fiber arm to screw through. The part is designed to be printed on its side. The design needs to be reworked as the single-hole point of connection doesn’t provide adequate resistance to rotation and the plastic threads holding it to the UAV wear out too quickly.

![Figure 3.6: Render of the UAV leg, with the screw-in point on top](image)

**Battery Holder**  The holder is designed to friction fit the battery in place; this allows for very easy removal to recharge the battery or swap it out for a spare. During normal flight operation, the friction forces are enough to hold the battery stationary and centered. In a crash condition, the battery can be expected to shift and will need to be re-centered before flying can resume. The four inner holes are where the ESC screws into the holder, the ESC does not directly mount to the frame. The battery holder mounts to the frame using 9mm standoffs that screw in on the four outer holes. Its designed to be printed on its side so the large central gap is along the Z-axis.

![Figure 3.7: Render of the battery holder](image)
Currently, the battery is held with the width being vertical, next generation of this part will mount it with the width being horizontal so that the CG is shifted higher.

**9mm Standoff** This part is needed due to the limitations of FDM 3D printing. It is a 9mm high cylinder with a hole through it. It gets screwed into from the upper side of the frame and the lower side of the battery holder.

![Figure 3.8: Render of the 9mm standoff](image)

**X Frame Bottom Support** This part is the lower section that holds the carbon fiber UAV arms. The center plus shape is needed because the Raspberry Pi required the frame to be slightly larger. It serves as something for the arms to compress into when they are bent. It also helps center them during assembly. It is quite a tight fit and the center plus is always in a compression. It is designed to be printed as shown with the plus shape facing up.

![Figure 3.9: Render of the X frame bottom support](image)

**X Frame Upper Support and Flight Controller Enclosure** The upper section of the X frame support serves to clamp down on the UAV arms, to protect the PixRacer from debris and the strong prop wash created by the propellers and to connect to the Raspberry Pi 3 mount. The rear opening is the smallest now as it only serves to let the PWM output connect to the ESC. The other connections are left exposed until the connections are finalized, at which point the holes will be sized and closed off appropriately. The part is designed to be printed as shown with the hollow shell vertical.

Going forward it would be convenient to be able to 3D print the UAV arms themselves; this would allow the lower X frame support and this upper X frame support to be combined and printed as one unit. This shift would drastically reduce build complexity and save money by no longer having to buy the arms. The limitation to this is part rigidity; nylon is much less stiff than the carbon fiber arms. A possible way around this would be to print the part hollow and to fill it with an epoxy and fiberglass composite.

13
Raspberry Pi 3 Mount The Raspberry Pi 3 mount serves the dual purpose of both securing the RPi and providing a lid to the upper frame support. The RPi mounts onto the slightly raised holes. The flush holes are where the mount screws into the upper frame support. The highest holes reach above the RPi and hold the Pozyx tag mount. Original iterations of this design had the RPi holes and the holes securing to the top mount overlapping. The overlap was done to reduce the complexity of the design and the total screw count. In the next iteration, the overlap was removed, and the CG of the RPi was centered above the center axis of the UAV. Testing this showed it was too extreme of a shift. The protruding board started to interfere with the aerodynamics of the front propellers causing the UAV to wobble when traveling forwards and backward. The next and most recent iteration shifted the board to be simply centered based on where the center of the rectangular board was. Experimentation showed this to be the best position so far and eliminated any wobble when performing maneuvers. The part is designed to be printed as shown with the cylinders pointed upwards.

Further iterations of this part may see the screw holes shifting places to be further out, or the addition of more screws better securing it to the upper frame support. At the moment now no changes are expected.

Pozyx Holder The Pozyx was harder to mount because unlike all the other parts, it has no screw holes. Instead, the tag would need to be clasped onto, and the holder would need to have the screw holes. The Pozyx tags surface complexity made two iterations of the design necessary to ensure it closed evenly on the device and didn’t catch on any of its surface mount components or its connectors. This task would have been very easy with a 3D object scanner, but without one, a caliper was used instead. The screw holes are thru-holes which line up with the top cylinder holes on the Raspberry Pi 3 mount. The bottom holder is designed to be printed as shown, the top holder needs to be flipped upside down for proper printing.

Part Printing Parts for the UAV frame and mounting components were 3D printed out of Taulman Alloy 910, a nylon filament. This filament was chosen because it is nylon, which allows for bolts to be directly screwed into it and for its high durability and stiffness allowing for the parts to withstand the forces experienced during a crash and to reduce the warping the frame can experience.
during flight. The parts were printed using a Prusa Mk2S with a 0.4mm nozzle head and a 0.2mm layer height. The print bed was heated to 60°C, the nozzle to 260°C, and the ambient temperature to 40°C.

**Receiver Switch**  During the original receiver choice, insufficient attention was given to transmitter selection. Because of this when it became time to find a DSMX transmitter no ideal option was determined. Instead, the option to buy a different receiver that still had multi-frequency operation and an S.Bus connection was chosen. The new receiver chosen is the FrSky XSR 2.4Ghz 16Ch for $17, the transmitter chosen is the FrSky Taranis Q x7 2.4Ghz 16Ch for $120.

**Initial Design Analysis**

**Vibration and Stability Issues**  Initial tests on UAV stability showed the flight controller was experiencing severe vibration causing the gyros to drift during flight. This noise can be seen in figure XXC. The rate gyros are responsible for the UAV remembering which way level was defined to be when they start to drift the UAV rapidly becomes impossible to control as it would think its level even if it has a 10 error from level. An accelerometer is not enough to know which way down is accurately because such small angle errors are very difficult to detect using the Z-axis measure from the accelerometer.

One of the best solutions to reduce noise, as recommended by ArduCopter, was to soft mount the flight controller using gel. The gel chosen was Kyosho Z8006 Zeal Vibration Absorption Sheet is adhesive on both sides of the gel. The sheet was cut into four squares which adhere to four sides of the flight controller mount. Because the flight controller is designed to be screwed into its mounting position a new mount had to be designed for it, shown in Figure 3.13 along with the gel. The four flat sides of the mount are where the gel is attached to the underside. Vibe measurements are a processed data of the accelerometer measurements. From ArduCopter documentation on the calculation used to measuring vibration:

![Figure 3.12: Render of the X frame Pozyx holder top and bottom sections](image)

![Figure 3.13: Kyosho Z8006 Zeal Vibration Absorption Sheet and the PixRacer flight controller mount.](image)
Capture the raw x, y and z accelerometer values from the primary IMU high-pass filter the raw values at 5hz to remove the vehicles movement and create a accel_vibe_floor for x, y and z axis. Calculate the difference between the latest accel values and the accel_vibe_floor. Square the above differences, filter at 2hz and then calculate the square root (for x, y and z). These final three values are what appear in the VIBE msgs VibeX, Y and Z fields.

The addition of the damping sheet eliminated the gyros accumulating error and drastically reduced the vibration noise the flight controller was measuring. However, the Vibe X,Y,Z measures are still higher than the <1 values recommended by ArduCopter documentation. Additional work to further reduce these values will be needed, but they are now at a serviceable point.

Figure 3.14: Without gel, dampening vibration issues seen by high VIBE. Vibe measures and VIBE. Clip values being nonzero.

Figure 3.15: With gel, Vibration Measurements and VIBE: Clip being zero

**Power Analysis**  The Power Analysis of the UAV was done by measuring how much current the motors consumed while the UAV was stationary using a Fluke 87-V Digital Multimeter. Throttle values ranged from 0-100%. The 0% throttle corresponded to the base power the flight controller would use. This base value was subtracted from the current used at throttle to determine how much power the motors consumed. Both an individual motor was tested as well as all four at once and their values averaged together. The four at once current draw for 80%-100% throttle exceeded the measurement range of the multimeter and had to be extrapolated from earlier data. The extrapolation used is a second order polynomial, the relationship makes sense because the torque required to spin the blades is dependent on the air drag of the propellers which is a second
order relationship. The linear term represents the resistive losses of the motor as well as the thrust generated. The four motor current draw is lower than the single motor current draw, this is to be expected as slight voltage sags under load will decrease current output, combining the values results in a conservative estimate.

The current draw data was then compared with throttle log data collected while the UAV hovered at a weight of 414g. The mean throttle value of 1415, corresponds to an average power the motors would consume in a hover, 3.9A. The value was then linearly scaled to an expected final power draw for the 500g weight with a Pozyx and Raspberry Pi 3. It came out to consume 4.71 amps for a hover. The Raspberry Pi 3 and the Pozyx tag were expected to use two amps, maximum, at 5V. The average operating voltage of the 3S HVLi battery will be 11.4V. The power system used to convert the voltage is assumed to have a conversion efficiency of 85%, which, based on previous experience with switching mode power supplies, is a reasonable estimation. However, actual measurements still need to be done. The voltage difference ratio, 2.28, and conversion efficiency gives a predicted current draw of 1.032 amps.

The total expected power draw of the whole system at hover is 7.745 amps. The battery is advertised as 1800 mAh, but the measured total capacity is slightly higher at 1943 mAh. To prolong the life of the battery it will only be drained to 85% capacity, giving an effective capacity of 1652 mAh. This gives the expected hover flight duration of 1034 seconds or 17.24 min.

<table>
<thead>
<tr>
<th>Throttle Value %</th>
<th>Current Draw (A)</th>
<th>Current - Base (A)</th>
<th>Extrapolated to Four</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.160</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>10</td>
<td>0.230</td>
<td>0.070</td>
<td>0.280</td>
</tr>
<tr>
<td>20</td>
<td>0.360</td>
<td>0.200</td>
<td>0.800</td>
</tr>
<tr>
<td>30</td>
<td>0.699</td>
<td>0.539</td>
<td>2.156</td>
</tr>
<tr>
<td>40</td>
<td>1.105</td>
<td>0.945</td>
<td>3.760</td>
</tr>
<tr>
<td>50</td>
<td>1.703</td>
<td>1.543</td>
<td>6.172</td>
</tr>
<tr>
<td>60</td>
<td>2.500</td>
<td>2.340</td>
<td>9.360</td>
</tr>
<tr>
<td>70</td>
<td>3.455</td>
<td>3.295</td>
<td>13.18</td>
</tr>
<tr>
<td>80</td>
<td>4.600</td>
<td>4.440</td>
<td>17.76</td>
</tr>
<tr>
<td>90</td>
<td>5.86</td>
<td>5.70</td>
<td>22.8</td>
</tr>
<tr>
<td>100</td>
<td>7.24</td>
<td>7.08</td>
<td>28.32</td>
</tr>
</tbody>
</table>

Table 3.1: Single Motor Test of Motor A

<table>
<thead>
<tr>
<th>Throttle Value %</th>
<th>Current Draw All (A)</th>
<th>Current - Base (A)</th>
<th>Extrapolated to One</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.160</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>10</td>
<td>0.482</td>
<td>0.322</td>
<td>0.0805</td>
</tr>
<tr>
<td>20</td>
<td>1.120</td>
<td>0.960</td>
<td>0.240</td>
</tr>
<tr>
<td>30</td>
<td>2.256</td>
<td>2.096</td>
<td>0.524</td>
</tr>
<tr>
<td>40</td>
<td>3.763</td>
<td>3.603</td>
<td>0.901</td>
</tr>
<tr>
<td>50</td>
<td>5.71</td>
<td>5.55</td>
<td>1.388</td>
</tr>
<tr>
<td>60</td>
<td>8.27</td>
<td>8.11</td>
<td>2.075</td>
</tr>
<tr>
<td>70</td>
<td>11.16</td>
<td>11.00</td>
<td>2.75</td>
</tr>
<tr>
<td>80*</td>
<td>14.464*</td>
<td>14.304*</td>
<td>3.576*</td>
</tr>
<tr>
<td>90*</td>
<td>18.232*</td>
<td>18.072*</td>
<td>4.518*</td>
</tr>
<tr>
<td>100*</td>
<td>22.44*</td>
<td>22.28*</td>
<td>5.57*</td>
</tr>
</tbody>
</table>

Table 3.2: Motor Test - All *extrapolated data from polynomial equation for 0-70% throttle
Figure 3.16: Single Motor Draw

Figure 3.17: All motor Current Draw without the extrapolated data of 80%, 90%, 100% throttle
<table>
<thead>
<tr>
<th>Throttle Value %</th>
<th>Current Draw All - Base(A)</th>
<th>Throttle PWM Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>986</td>
</tr>
<tr>
<td>10</td>
<td>0.301</td>
<td>1084</td>
</tr>
<tr>
<td>20</td>
<td>0.480</td>
<td>1186</td>
</tr>
<tr>
<td>30</td>
<td>2.126</td>
<td>1289</td>
</tr>
<tr>
<td>40</td>
<td>3.6915</td>
<td>1391</td>
</tr>
<tr>
<td>50</td>
<td>5.861</td>
<td>1494</td>
</tr>
<tr>
<td>60</td>
<td>8.735</td>
<td>1596</td>
</tr>
<tr>
<td>70</td>
<td>12.09</td>
<td>1698</td>
</tr>
<tr>
<td>80*</td>
<td>16.03*</td>
<td>1801</td>
</tr>
<tr>
<td>90*</td>
<td>20.44*</td>
<td>1903</td>
</tr>
<tr>
<td>100*</td>
<td>25.30*</td>
<td>2006</td>
</tr>
</tbody>
</table>

Table 3.3: Combined Motor Test Data - All *Data using extrapolated all motor current draw.

Figure 3.18: Current draw minus the base.
**CG Analysis**  The center of gravity of the UAV was determined by hanging the UAV and taking pictures to see how it aligned with the string it was suspended from. It was hung from its back left and back right arms. The C.G. in the horizontal plane was aligned with the center of the flight controller but had 16mm offset in the vertical plane. To correct for this the ArduCopter parameters INS_POS1_Z and INS_POS2_Z were set to +0.016m. It is positive because the Z axis is pointed downwards on the craft. From ardupilot.org, *the compensation is only partial because ArduPilot can correct the vehicles velocity and position estimate but it does not correct the acceleration estimate.* A goal over the winter break is to rotate the battery, which will shift the C.G. slightly higher on the craft reducing the Z offset.

![Figure 3.19: The Z-direction and Y-X-direction left and right CG offsets](image)

![Figure 3.20: X-Y offset images superimposed over each other](image)
3.2.2 Intermediate Designs

Thrust Analysis of Ducted Propeller

To expand the initial design considerations and attempt to increase hover time the effect of adding a duct to the propellers was considered. The benefits of adding a duct would be increased hover performance by decreasing wing tip vorticity losses, and increased safety as the entire propeller would be enclosed. The drawbacks would be increased design complexity, increased weight, and worse crosswind performance. To determine if the addition of a propeller duct was beneficial or not, static thrust tests were conducted of both a free propeller and a ducted propeller. These experimental setups are shown in Figures 3.21 and 3.22.

![Figure 3.21: Propeller thrust test configuration without the addition of a duct.](image1)

![Figure 3.22: Propeller thrust test configuration with the addition of a duct.](image2)

The design of the test fixture was of an L-shaped arm with each leg of equal length (200mm) and a pin at the bend to rotate freely around. The propeller was mounted horizontally so that as the propeller thrusts it forces the other end of the L into a scale to measure force. The scale was a AMIR brand 500g kitchen scale with a 0.01g resolution. The motor was controlled using the flight controller by commanding thrusts ranging from 10 to 100 percent. To measure current draw a Fluke
87V multimeter was wired into the battery connection to the entire UAV. This setup measured draw of the flight controller as well so to accommodate for that an offset value (42 mA) was subtracted from the measured current draw.

The duct was designed to have a 0.25mm gap between the propeller and shroud, an angle of 5, and a total length of 80mm. This design was decided on based upon previous work by Jason L. Pereira [20], and can be seen in Figure 3.23.

![Figure 3.23: Propeller duct cross-section dimensions.](image)

Ultimately it was decided that given our thrust requirements (125g per motor) that adding a propeller shroud while it would increase efficiency (14% at 125g), wouldn’t increase it enough to offset the added weight, complexity and increased crosswind sensitivity. A chart of the current draw vs thrust is shown in Figure 3.24.

![Figure 3.24: Current draw of the ducted (blue) and free (orange) propellers compared against measured thrust.](image)

**Range Finder**

Initial localization designs had the UAVs Z axis height being based entirely on the Pozyx system and the flight controllers internal barometer. While this worked in the short term a better system
was needed because the Pozyx vertical accuracy was poor due anchor placement limitations (they were all within a 1 meter plane with each other) and the barometers measured altitude would drift overtime. To solve these we added a downwards facing ultrasonic range finder. The range finder used was the MB1043 HRLV-MaxSonar-EZ4, shown in Figure 3.25. It was chosen because it would be directly compatible with the Raspberry Pi UART, had the narrowest beam pattern, and provided fast readings with 1mm resolution. A narrow beam pattern was chosen because only the floor is to be sensed and having a wide beam would increase the chances of it measuring off angle objects like a person standing nearby. Additionally, a narrower beam pattern would decrease interference that would arise from multiple UAVs ranging concurrently.

![Figure 3.25: MB1043 HRLV-MaxSonar-EZ4 mounted on the UAV with a smoothing capacitor soldered on to it.](image)

**Power Issues**

One major issue plaguing the initial designs was that using the remote to control the drone would result in the Pozyx loosing almost all rangefinder beacons and giving erratic positions. The initial efforts to solving this issue revolved around checking to make sure the Pozyx had enough power and that the remote controls signal wouldnt be interfering with the connection. This problem went unsolved until the range finder was added, and similar issues occurred with it. It would range correctly when the remote was off but once it was activated it would report a range error. This provided enough additional information to rule out most of the potential issues and narrow it down to not be related to the transmitter but be related to the drones receiver.

![Figure 3.26: DROK LM2596 Buck Converter located on the rear of the drone.](image)
What happened is that when the transmitter was turned on, the receiver would start drawing power irregularly and cause large voltage swings and transients in the 5V line. When off there was an AC noise of 30mV in the 5v line, when on the noise would rise to 80mV. Initial efforts to solve this included adding a large 2200uF smoothing capacitor to the Pozyx and a 100uF smoothing capacitor to the rangefinder. The addition of the capacitors did improve both the rangefinder and the Pozyx performance it did not do enough to solve all issues. The final solution was to add a separate power converter for the receiver to fully isolate its noise. The additional power converter chosen was a DROK LM2596 Buck Converter rated up to 3A, shown in Figure 3.26. The receiver was placed on the original APM2.1 power converter and the flight controller, Raspberry Pi, Pozyx and rangefinder were placed on the higher quality DROK.

Frame Iterations

In order to further reduce costs and increase design freedom the drone frame design was reconsidered. Several iterations later, a pure 3D printed nylon with carbonfiber particul reinforcement, a carbonfiber particul reinforcement reinforced with epoxy, a PLA and steel rod combined with epoxy, a corbonfiber and steel rod combined with epoxy. and finally cut carbonfiber sheet and 3D printed nylon frame epoxied together.

The design considerations for the frame included cost, stiffness, cost, durability, and versatility. Ultimately the carbon fiber rectangles cut out from a large sheet epoxied together 3D printed nylon frame proved the best stiffness and resulted in minimal costs. These final frames cost only 7$ per. A reduction from the original cost of $30. The frame is also lighter weighing only 50 g down from 99 g.

3.2.3 Final Design

After much analysis and several iterations, the final design for the UAV sensor vehicles has completely alternate components than the original design, and is shown below.

Figure 3.27: The final assembled UAV design with all components.
Main Body

The main body of the drone consisted of two sections. An upper section which served as the housing for the flight controller and the mount for the Raspberry Pi, and a lower section which held the arms and served as the structural core of the drone. Both parts were 3D printed in Taulman Alloy 910 and then epoxied together for structural rigidity. The process can be seen in Figure 3.28.

Figure 3.28: Images of the drone frame during assembly, before, during and after epoxying the Raspberry Pi mount to the arms.
Flight Controller Mount

For the new design the flight controller mount was also redesigned to allow the flight controller to sit closer to the frame. This was done by moving the vibration dampening pads from below the flight controller to being around it. The shift allowed the flight controller to sit 5mm lower and allowed the Pi to also be mounted lower as well. The part was 3D printed in Taulman Alloy 910, and is shown in Figure 3.29 below.

![3D printed mount for the flight controller to sit.](image)

ESC Mount and Lower Frame Hardpoint

In the new design the ESC also shifted its mounting position. In the original design the ESC was mounted on the battery holder. This was a challenge because it had to be screwed in place before the holder was added to the frame. This made the drone difficult to assemble because the motors would have to be soldered after the battery mount was added. The new frame allowed for greater mounting options so the ESC was shifted to screw into the main body, as seen in Figure 3.30. The ESC mount was also combined with a hardpoint where the lower frame can be attached. The mount/hardpoint piece is connected to the main body with four screws. Its not epoxied in place so that if a crash occurs it will break and help absorb the force of the impact.

![ESC mount and lower frame hardpoint. The smaller risers in the middle are where the ESC mounts. The outer larger risers are where the Lower Frame connects. The small holes are the screw in points to connect it with the main body.](image)
Lower Frame

The lower frame serves four functions. It holds the battery, it mounts the rangefinder and camera, and it is the surface the drone lands on. Due to the complex shape of the design it had to be printed in two parts and joined in post processing. This was done by using a soldering iron at 450F to reheat the plastic and melt the sections together. Images of the frame before and after this soldering are shown in Figure 3.31. The large ovals cut into the top and bottom are to save weight.

![Figure 3.31: The lower frame both before it was melted together, and afterwards holding all of the components.](image)

Frame Arms

The frame arms are the most time-consuming section of the drone to make. First, rectangles are cut from flat carbon fiber sheet using a Dremel to the approximate thickness of 5.5 mm, as seen in Figure 3.32. They are then further ground down with the Dremel to the exact 5mm width. A shop-vac was setup below the jig to keep it tidy and full Gas masks were worn during grinding periods. Width of the grinding and cutting was controlled by repositioning the Dremel head relative to the jig wall
the carbon fiber was braced against. After the arms are the correct thickness they are cut in half to be 100 mm.

**Motor Mounts**

There are two variations of the motor mount, one for clockwise one for counterclockwise. They are functionally identical except that the angles where they insert onto the frame arms are slightly different. The mounts are not actually secured onto the frame in anyway, they are held in place by friction alone. This was done so that in a crash the motors could break off their mounts and be replaced without damaging the main body of the drone. As with the rest of the drone parts it was printed in Taulman Alloy 910.

### 3.3 Sensing Mechanism

In order to optimize computations, it seemed convenient to perform the image processing near the source of the data, utilizing the Raspberry Pi on-board each sensor vehicle. The first approach to image sensing consisted of developing a script that returns the average RGB color of a cropped central section of the original image. The position and size of the section to be secluded within the frame are determined by the input parameters offset and size. The image is then cropped using the Image module of the Python Imaging Library. The cropped section is then converted from an image to an array using the array function from the NumPy library, so that each entry in the array contains the three RGB values corresponding to each pixel in the image. Finally, the red, green and blue values are averaged respectively, and the resulting RGB value is returned as the average color of the image. This color is calibrated to represent a particular threat value, which is then fed into the IPAS algorithm for sensor reconfiguration and path trajectory adjustments.

As mentioned in the threat field representation section, the threat field colormap was selected to
be a perceptually sequential colormap named plasma from the matplotlib library in Python. This colormap contains 256 colors. In order to perform adequate image calibration, a python script was written to translate the average color sensed by the pi camera into a threat value that could be fed into the IPAS algorithm. The general idea behind this translation script is to find the best match for the sensed color within the list of 256 colors of the colormap and return the index of the color in the colormap list as the threat value. Thus, there is a total of 256 threat levels ranging from 0 (lowest threat value represented by dark blue in the plasma colormap) to 255 (highest threat value represented by bright yellow in the plasma colormap).

One challenge in the practical implementation of this translation script is to deal with a sensed color provided by the pi camera that does not exactly match any of the 256 colors in the colormap. This is very likely due to the threat field being projected at an angle (which affects the brightness of the projected image) and other room conditions and camera settings that produce a picture which colors are not exactly the same as the true image. To solve this issue, a similarity function named best_match was developed that modeled color similarity as an euclidean distance between two points. Therefore, in the script, once a color is received from the camera, it is passed to the best_match function along with the 256 colors from the colormap. The euclidean distance is then calculated between this sensed color and each of the colors in the colormap. All of these distances are stored and then sorted in increasing order. The best_match function returns the first item in the sorted list of distances as the best match in the colormap and thus, a threat value can be calculated for a sensed color that does not exactly match any color in the map.

3.4 Localization

3.4.1 Pozyx Set Up

In order to enable optimal indoor localization adhering to Pozyx’s suggested set up, the Pozyx anchors were arranged following the diagram in Figure 3.33. The hexadecimal numbers correspond to the unique IDs of each of the four anchors, and each h indicates the height at which each anchor was mounted. Each anchor was placed at a different height ranging from one to two and a half meters, and mounted on a wooden structure to avoid metallic interference, shown in Figure 3.34.

![Figure 3.33: Layout of the Pozyx anchors](image)

This layout allowed the Pozyx tags to be tracked in a field of approximately 8x16 m². This is more than adequate for the demonstration considering the projected field has dimensions of 6x7 m². It also proves beneficial to have a smaller threat field than tracking range since the adverse effect on location accuracy when the tags get closer in proximity to the anchors has been directly observed through testing.

3.4.2 Pozyx Positioning

To use the Pozyx system, Python code was written to facilitate communication between the Pozyx tag and the PixRacer flight controllers on each UAV. The code needs to be initialized with the
measured locations of the anchors, which are found by assuming 0x6110 is the origin, and then measuring the distance to each anchor with a laser range finder. When running, the code continuously loops requesting the ranges to each anchor from the tag, and the current estimated position is sent to the flight controller. The Pozyx tag is connected via USB and the data received from it is fed into the PixRacer on the TELEM2 port using the GPIO pins on the Raspberry Pi. Since ArduCopter version 3.5, directly feeding in Pozyx data has been supported natively without having to modify the data or convert it to another format. However, because this feature is new, arming the UAV with only the Pozyx and not a GPS is still considered a safety hazard and therefore is not allowed by default. In order to work around this, since the testing is occurring indoors where GPS is not as accurate, the code was further developed to alter the data returned by Pozyx into GPS location data that can be fed through MavProxy to the PixRacer. This provides a GPS lock, allowing flight without a physical GPS.

3.4.3 Improving Pozyx Accuracy

Due to the implicit complications of a physical system, localization data acquired from the Pozyx system is inherently noisy. For the UAVs, this is not an issue as ArduCopter v3.5+ has support for the Pozyx natively and integrates it into its own internal Extended Kalman Filter. However, for the UGVs such as the UGV sensor and the Actor, no such Kalman Filter exists. Therefore, in order to reduce this noise for the UGVs, a Kalman Filter was developed to improve the accuracy of the system.

In order to reduce development time, the FilterPy library [16], along with its accompanying textbook: Kalman and Bayesian Filters in Python [17], were employed so that only the state transition matrix and the measurement noise matrix needed to be specified. The state variables were chosen to be the x and y position of the UGV and their first and second derivatives along with the heading \( \theta \) with respect to the x-axis and its first derivative:

\[
X = [x \quad \dot{x} \quad \ddot{x} \quad y \quad \dot{y} \quad \ddot{y} \quad \theta \quad \dot{\theta}]^T
\]
For these, the state transition matrix is defined for any change in time, \( \Delta t \):

\[
F = \begin{bmatrix}
1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 & 0 \\
0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \Delta t & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

Measurements for \( x, y, \ddot{x}, \ddot{y}, \) and \( \theta \) can be read directly from the Pozyx localization data and its internal sensors. For the remaining three state variables, \( \dot{x}, \dot{y}, \) and \( \dot{\theta} \), the encoders on the wheels are employed to compute the change in position and heading using the following differential drive equations and dividing by the change in time [18]:

\[
\begin{bmatrix}
\Delta x \\
\Delta y \\
\Delta \theta \\
\end{bmatrix} = \begin{bmatrix}
cos(\omega \Delta t) & -\sin(\omega \Delta t) & 0 \\
\sin(\omega \Delta t) & \cos(\omega \Delta t) & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \begin{bmatrix}
-ICC_x \\
-ICC_y \\
\end{bmatrix} + \begin{bmatrix}
ICC_x \\
ICC_y \\
\end{bmatrix} \frac{\omega \Delta t}{\Delta t}
\]

where

\[
R = l/2(v_r + v_l)/(v_r - v_l) \\
\omega \Delta t = (v_r - v_l)\Delta t/l \\
ICC_x = -R \sin(\theta) \\
ICC_y = R \cos(\theta)
\]

The values for \( v_l \) and \( v_r \) are the linear speed of the left and right wheels, respectively, and are internally maintained by the handling code for encoders. They are updated whenever the callback is triggered by encoder sensors.

As for the measurement noise matrix, the advertised precision of each system was chosen as the starting noise value (e.g. 10cm for \( x \) & \( y \), 36° for the encoders) and then they are tuned through trial and error for the particular current environment.

### 3.4.4 Implementing Pozyx on Multiple Vehicles

Unlike a positioning system like GPS, the Pozyx does not use a broadcast mechanism where a device that wants to position only listens for regularly spaced broadcasts from known positions. Instead, each tag communicates directly with each anchor to calculate its own position. This bi-directional flow of information allows for higher accuracy but comes at a cost: only one tag can position itself at a time. In the Pozyx documentation this is noted as normally not an issue because it was assumed by the developers that each tag would be controlled from a master device which would simply loop over all the tags and position them one at a time. However, for our purposes, each vehicle should be able to position itself without requiring a master device to be online.

In order for each vehicle to be able to position itself, clashes need to be avoided. This means making sure that only one vehicle is positioning at a time without requiring a master device to deviate out instructions. To solve this problem, a network-based linked list solution was developed. Because each vehicle on the network falls into a given fixed IP range and has a static IP, when a vehicle comes online, it is able to scan the network and find another vehicle if another one is online. By incrementing its own IP address until the maximum is hit and then starting over from the minimum, all addresses in the known space can be queried. Upon finding another vehicle, it can connect to it and inform it to start positioning whenever it has finished. Since all vehicles will have done this upon coming online, the vehicle with the highest IP address will have connected to...
the one with the lowest IP address. The one with the lowest IP address then knows that it can go
first (because it is the only one that was connected to by a greater IP address than itself) and starts
positioning. When it is done, it informs the next to start and then waits for the signal to come back
around the loop to itself before positioning again.

This system has additional features built in to increase its robustness. In the downtime between
positioning, each vehicle reads its internal sensors and stores their state for usage elsewhere and
pings its connected vehicle to make sure it is still online. Should one vehicle leave, this is recognized
immediately and the upon failing to ping its connected vehicle, the search for a new connection
starts over. Connections are accepted greedily, meaning that if a vehicle attempts to connect to a
vehicle which already has a connection, the old connection is closed and dropped and the new one
is accepted. This means that the vehicle which left will be dropped in favor of a vehicle that was
cconnected to the one which dropped. For example, in a system of three vehicles where vehicle 1
is connected to vehicle 2 which is connected to vehicle 3 which is connected back to vehicle 1, if
vehicle 2 leaves the system, vehicle 1 will connect to vehicle 3 which will accept the connection.
This system also works in the opposite scenario. Should a vehicle come online after the others have
started positioning, it will seek out a connection. Using the previous example, if vehicle 2 comes
back online, it would find and connect to vehicle 3 which would close the backward connection to
vehicle 1. When vehicle 1 finds its connection closed, it will seek out a new connection and find
vehicle 2 first, restoring vehicle 2 to its former place in the linked list.

3.5 Implementing the IPAS Algorithm

3.5.1 Communications

In order to communicate between the vehicles, the proposed HTTP based system was developed
where the actor vehicle is the master, making requests for information to each sensing vehicle using
HTTP over Wi-Fi. For physical equipment, we are using the Wi-Fi module built into the Raspberry
Pi 3 and a Netgear Nighthawk AC1750 as the router. The router provides an isolated (non-Internet
connected) network that every device can connect to reliably. Once online, each device is assigned
a static IP address so that the actor can communicate directly with them.

On the software side, requests will be sent from the actor to the Raspberry Pis on each sensing
vehicle using the Python Requests library. The Nginx web server on each sensor will receive the
request and pass it up to Gunicorn, which will spawn a Python process running the Falcon web
framework to handle the request. The actor can use this web service to request image data and
threat values, as well as instruct movement to positions. Movement is accomplished using DroneKit-
Python for the UAVs and the custom Robot-Navigation server for the UGVs. More testing needs
to be performed on the position instructions at this moment.

3.5.2 Sensor Relocation Algorithm

An essential aspect of the IPAS algorithm that had not been previously addressed is the time and cost
of repositioning the sensor vehicles after each iteration of sensing and planning. This encompasses
the coordinated movement of multiple UAVs, avoiding collision or unnecessary time and energy
expenses. The sensor relocation algorithm needed consists of two parts: sensor assignment (where
the algorithm determines where the new location for each sensor will be), and path planning (where
the path to each new sensor location is generated).

Sensor Assignment

Sensor assignment is determined by a greedy algorithm, where sensors and new sensor locations
are numbered arbitrarily one through \( n \) and mapped to each other. If any two paths cross, the
destinations of the sensors are swapped. For example, if sensor one crosses paths with sensor four, then sensor one will now go to the new location four instead, and vice versa. This process continues until there are no intersections.

Path Planning

To generate a path between sensors and new sensor locations, the start and end points of each path are fed into matrices that generate a cubic polynomial trajectory. A cubic polynomial trajectory allows for the sensor trajectory to have initial and final velocities of zero in order to ensure a smooth flight path.

\[
q(t_0) = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3
\]

\[
v(t_0) = a_1 + 2a_2 t_0 + 3a_3 t_0^2
\]

\[
q(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3
\]

\[
v(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2
\]

\[
\begin{bmatrix}
1 & t_0 & t_0^2 & t_0^3 \\
0 & 1 & 2t_0 & 3t_0^2 \\
1 & t_f & t_f^2 & t_f^3 \\
0 & 1 & 2t_f & 3t_f^2
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3
\end{bmatrix}
= 
\begin{bmatrix}
q_0 \\
v_0 \\
q_f \\
v_f
\end{bmatrix}
\] (3.2)

Solving this system of equations gives the position and velocity as functions of time of a straight line trajectory. These are converted into waypoints for sensor relocation. This system of sensor reassignment naturally avoids collisions because in each iteration, we ensure that new positions are far enough apart to avoid collision. Since no paths intersect, sensors will not collide unless they are unable to follow the path.

Alternative Relocation

An alternative method for sensor relocation is used in the case that a greedy sensor assignment is not viable. If for any reason a certain sensor had to be placed in a certain position, or there are certain obstacles that sensors must avoid, we take a different approach in which sensor assignment is arbitrary. A* path planning is used to generate a path for each sensor to its destination using a simulated field with Gaussian peaks centered at each of the other sensors starting locations. This produces a relatively simple path for each sensor. The next step is to sort the paths by length. The sensor with the shortest path re-plans its trajectory with a moving threat field of Gaussian peaks centered at the expected location of each sensor at that given time. Then the second sensor re-plans its path with the same assumptions, only it expects the first sensor to move along the new path. This process repeats until the final path is generated.

3.5.3 Actor Vehicle

Two of the DFRobot Arduino kits were purchased, one for the actor vehicle and one to test as a sensor UGV. Although the vehicle is intended for 4WD, it was deemed more fitting to use just two drive motors with a third castor ball in the rear to enable steering without having to rely on wheel skidding.

Wheel velocity was determined using the time between encoder rising and falling edge signals. By dividing the angle between encoder edges by the time between signal edges, an estimate of the angular velocity of the wheel could be determined. Encoder feedback was used to form a closed loop control system with the wheels to reduce angular velocity settling time. Vehicle navigation between waypoints is performed by simple proportional movements, ceasing motion and setting desired motor speed to zero, or proceeding to the next waypoint upon crossing the desired proximity threshold. Using the Pozzyx Kalman Filter described in Section 3.3.3, the Pozzyx localization data can be combined with the wheel encoder data to improve the accuracy of the proximity threshold and arrive at the proper location.
Chapter 4

Results and Discussion

4.1 Field Projection

4.1.1 Final Projection Set Up

Figure 4.1: Final threat field set up, utilizing four projectors mounted on stands atop tables, and the white paper background for enhanced image and data quality.

The final threat field set up can be seen in Figure 4.1, including two projectors and the white background, displaying an image fed into and split using the Python script. Three of the four Pozyx anchors can be seen in the corners as well. Figure 4.2 shows the completed mounting brackets used to secure the projectors to the speaker stands.

4.1.2 Sensor Readings

During the system development stage, a colormap from the matplotlib python library was used to represent our threat field. However, once we projected the images using two different projectors,
there were notable color differences between the two images as shown in Figure 4.3. Given that these were the only projectors available, an efficient solution was to replace the colormap in use for a grayscale to represent the threat field and hence eliminate, to an extent, inconsistency across projectors. The grayscale threat field shown on Figure 4.4 displays much less inconsistency even though the same set of projectors was still used. With this setup, a camera calibration test was performed to corroborate camera output and translation from color value to threat.

Figure 4.5b shows a frame captured by the picamera that corresponds to the red box on figure 4.5a, which represents the complete field. The camera interpreted the values in the frame as 100 for the white dot and 130 for the cropped dot in the center of Figure 4.5b. Given that our threat levels range from 0 to 256, these sensed values fell within this range and represent an accurate estimate
since, as analyzed from the original image Figure 4.5a the values captured are just in the middle of the color range (from black to white).

(a) Original threat field representation using a grayscale. Red box indicates portion taken by picamera.

(b) Image captured by picamera.
4.2 Pozyx Precision and Accuracy

4.2.1 Position Precision

In order to understand how accurate and precise the readings from the Pozyx system are, we ran tests on a stationary Pozyx tag located on a sensor UAV. The anchor layout is indicated in Figure 4.6.

The data that the Pozyx system returns consists of X, Y, and Z coordinates, as well as the individual ranges from the tag to each anchor. If the tag fails to range to one of the anchors, a quick error message is printed, and the Pozyx continues to range until connection is reestablished. Sample data from the accuracy test is shown below in Table 4.1.

<table>
<thead>
<tr>
<th>Time Stamp</th>
<th>X-Coord</th>
<th>Y-Coord</th>
<th>Z-Coord</th>
<th>Range 0</th>
<th>Fail 0</th>
<th>Range 5</th>
<th>Fail 5</th>
<th>Range E</th>
<th>Fail E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.13138559</td>
<td>3964</td>
<td>4557</td>
<td>371</td>
<td>6035</td>
<td>0</td>
<td>5524</td>
<td>0</td>
<td>4592</td>
<td>0</td>
</tr>
<tr>
<td>0.56537867</td>
<td>3956</td>
<td>4562</td>
<td>385</td>
<td>6077</td>
<td>0</td>
<td>5501</td>
<td>0</td>
<td>4601</td>
<td>0</td>
</tr>
<tr>
<td>0.72032022</td>
<td>3957</td>
<td>4560</td>
<td>376</td>
<td>6063</td>
<td>0</td>
<td>5497</td>
<td>0</td>
<td>4612</td>
<td>0</td>
</tr>
<tr>
<td>0.8806509</td>
<td>3958</td>
<td>4561</td>
<td>377</td>
<td>6082</td>
<td>0</td>
<td>5510</td>
<td>0</td>
<td>4635</td>
<td>0</td>
</tr>
<tr>
<td>1.03830075</td>
<td>3958</td>
<td>4562</td>
<td>372</td>
<td>6040</td>
<td>0</td>
<td>5496</td>
<td>0</td>
<td>4654</td>
<td>0</td>
</tr>
<tr>
<td>1.19668961</td>
<td>3957</td>
<td>4562</td>
<td>366</td>
<td>6035</td>
<td>0</td>
<td>5520</td>
<td>0</td>
<td>4644</td>
<td>0</td>
</tr>
<tr>
<td>1.34917212</td>
<td>3956</td>
<td>4561</td>
<td>364</td>
<td>6077</td>
<td>0</td>
<td>5464</td>
<td>0</td>
<td>4621</td>
<td>0</td>
</tr>
<tr>
<td>1.50166273</td>
<td>3957</td>
<td>4565</td>
<td>368</td>
<td>6035</td>
<td>0</td>
<td>5492</td>
<td>0</td>
<td>4598</td>
<td>0</td>
</tr>
<tr>
<td>1.65796161</td>
<td>3957</td>
<td>4564</td>
<td>371</td>
<td>6063</td>
<td>0</td>
<td>5492</td>
<td>0</td>
<td>4597</td>
<td>0</td>
</tr>
<tr>
<td>1.80948177</td>
<td>3956</td>
<td>4563</td>
<td>368</td>
<td>6026</td>
<td>0</td>
<td>5474</td>
<td>0</td>
<td>4616</td>
<td>0</td>
</tr>
<tr>
<td>1.96068239</td>
<td>3957</td>
<td>4562</td>
<td>371</td>
<td>6040</td>
<td>0</td>
<td>5469</td>
<td>0</td>
<td>4607</td>
<td>0</td>
</tr>
<tr>
<td>2.1168077</td>
<td>3961</td>
<td>4557</td>
<td>359</td>
<td>6059</td>
<td>0</td>
<td>5506</td>
<td>0</td>
<td>4598</td>
<td>0</td>
</tr>
<tr>
<td>2.2690661</td>
<td>3962</td>
<td>4556</td>
<td>362</td>
<td>6008</td>
<td>0</td>
<td>5497</td>
<td>0</td>
<td>4616</td>
<td>0</td>
</tr>
<tr>
<td>2.41629243</td>
<td>3966</td>
<td>4551</td>
<td>350</td>
<td>6082</td>
<td>0</td>
<td>5483</td>
<td>0</td>
<td>4588</td>
<td>0</td>
</tr>
<tr>
<td>2.56526351</td>
<td>3968</td>
<td>4550</td>
<td>351</td>
<td>6049</td>
<td>0</td>
<td>5481</td>
<td>0</td>
<td>4588</td>
<td>0</td>
</tr>
<tr>
<td>2.7140387</td>
<td>3973</td>
<td>4544</td>
<td>335</td>
<td>6091</td>
<td>0</td>
<td>5497</td>
<td>0</td>
<td>4611</td>
<td>0</td>
</tr>
<tr>
<td>2.86515784</td>
<td>3973</td>
<td>4544</td>
<td>336</td>
<td>6035</td>
<td>0</td>
<td>5492</td>
<td>0</td>
<td>4610</td>
<td>0</td>
</tr>
<tr>
<td>3.01404238</td>
<td>3971</td>
<td>4544</td>
<td>334</td>
<td>6087</td>
<td>0</td>
<td>5464</td>
<td>0</td>
<td>4620</td>
<td>0</td>
</tr>
<tr>
<td>3.16550183</td>
<td>3970</td>
<td>4544</td>
<td>339</td>
<td>6049</td>
<td>0</td>
<td>5539</td>
<td>0</td>
<td>6035</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1: Sample data from the Pozyx localization system

Table 4.1 represents sample data that is returned from the positioning system over a period of three seconds. The ranges and coordinates are measured in millimeters and the indication for failure to range to each anchor is indicated with the number 1; otherwise, 0 means the ranging was successful. As indicated on the last entry of the table, during second 3.16, ranging to anchor E failed. When a ranging failure occurs, the reported range to that anchor is 0. However, when such a failure occurs, X, Y, and Z coordinates are still reported in that entry.

Plotting individual coordinates against time provides a better view of all the measurements returned by the system in this accuracy test. The plots of each coordinate axis data X, Y, and Z, along with the mean and standard deviation are shown in Figures 4.7, 4.8, and 4.9 respectively.
Figure 4.7: Pozyx X-coordinate localization measurements in mm

Figure 4.8: Pozyx Y-coordinate localization measurements in mm

Figure 4.9: Pozyx Z-coordinate localization measurements in mm
In blue are the measurements during which ranging to every anchor was successful. The measurements during which there was at least one ranging failure are denoted in red. From these plots, it can be concluded that the highest standard deviation is produced in the Z-direction. The minimum measurement in this direction was 249 mm and the maximum was 402 mm, generating a difference of 153 mm. The reported Pozyx error by the manufacturer is of 100 mm, and therefore the Z maximum difference is still within a relatively acceptable range. However, these measurements may require additional filtering for optimal loitering and vehicle coordination once other UAV sensors are developed and flown in the same environment.

In the Y-direction, the minimum and maximum measurements were 4523 and 4569 mm, respectively. This produced a maximum difference between measurements of 46 mm. Similarly, in the X-direction, the minimum and maximum measurements were 3954 and 3996 mm, producing a maximum difference of 42 mm. Both of these variations are within the Pozyx reported error of 100 mm, and as evident over the course of the two minute test, the measurements remain reliably stable with only slight variations.

### 4.2.2 Position Accuracy

While the system has demonstrated itself to be fairly precise, when it comes to the accuracy of the system, we have run into issues. During the accuracy test, the Pozyx tag was located on top of the UAV, about 12.8 cm above the ground. As can be seen in 4.9, the Z-values tend to be reported much higher, often around 30 centimeters. As the anchors are moved farther apart, this effect is amplified where if the setup shown in Figure 3.33 is used, Pozyx reports the tag at somewhere between 50 cm and 100 cm above the ground, even when it is stationary on the floor. Also, when the tag is raised vertically, the scale of the Z-axis appears to be wrong. If the tag is raised to 2 meters, it reports being at about 2 meters despite saying it started off at about 1 meter, and is therefore not simply an offset. More research needs to be done into why the Z-axis appears to be less accurate than its X and Y counterparts.

### 4.2.3 Effect of Anchor Measurement Errors

Figure 4.10 shows the effects of improper measurement of the location of the anchors. This error is purely mathematical - that is, it does not take into account the range measurement noise that compounds with it. An error of 500 mm in the X-direction of one anchor results in a 156 mm error in measurement of Z-position, and an error in two anchors of 500 mm leads to a Z-position error of 624 mm. These errors also skew the scale of the Z-axis. An error of 500 mm in one anchor results in an altitude change of 1000 mm being read as an altitude change of 681 mm. This demonstrates the importance of starting with highly accurate positions for the anchors.

![Figure 4.10: Pozyx localization error due to error in anchor placement](image-url)
4.3 IPAS Algorithm

4.3.1 Sensor Relocation Algorithm

The diagrams in Figure 4.11 show the before and after greedy relocation algorithm, resulting in $n$ (in this case 30) paths that have no intersections. The cubic polynomial trajectory will allow for all sensors to traverse their paths in the same amount of time, as well as arrive with a zero velocity.

![Figure 4.11: Greedy relocation algorithm paths](image1)

The next figure shows an example of the output paths from the alternative relocation algorithm moving six sensors around the field. Sensors move from the circle to the X. Notice that the paths do cross, however, the timing is such that they do not meet. This is why the purple and green paths start in odd patterns, so that when the purple marker reaches the yellow path, the yellow sensor has already crossed, and when the green marker reaches purple, red, and yellow, they too have already gone. The algorithm safely moves sensors from arbitrary starting positions to arbitrary final positions without collision.

![Figure 4.12: Alternate relocation algorithm paths](image2)
To make the paths smoother, a method for waiting could be implemented. This would allow sensors to stay still for a period of time rather than move around unnecessarily as we see the green sensor do here. So far the issue with implementing this is that it greatly increases the computation time for the algorithm. The iteration shown above takes less than one second to complete, whereas including the ability for the sensors to wait takes on the order of hundreds of seconds.

4.3.2 IPAS Execution

At this current time, 3 executions of the IPAS algorithm have taken place. Due to issues with the position stability of the UAVs, the tests were completed by manually moving sensors from one location to another on 10x10 grid. Figures 4.13a and 4.13b show the results of the first two tests. The third test was performed with 3 sensors, however it had a technical issue in which one of the sensors’ cameras failed to record anything, resulting in failed a execution in which the path passed through threats since it could not see them.

Despite being limited to positioning simply ‘by eye’ and having a limited number of sensors, the results of the first two tests are extremely promising. They demonstrate that the algorithm indeed works, along with the communication protocols established to each sensor. More in-depth tests with records of recorded values and estimated threat fields still need to be performed, along with the usage of both the UAV and UGV sensors.

(a) IPAS test result with 1 sensors moved manually on a 10x10 grid. (b) IPAS test result with 2 sensors moved manually on a 10x10 grid.
Chapter 5

Localization Using Wireless Signals

5.1 Introduction

Radio ranging is a commonly used technique for localization of a device. For example, GPS functions using the time dilation from several satellite signals to determine its own position on Earth. A similar method is employed by the Pozyx system, where ranges are measured from the known positions of four anchors and the 3-D position is derived from those ranges. The common method used is to determine the ranges to known points and from there the task of localization becomes somewhat trivial, or at the very least well researched and documented. The location can be analytically determined as an intersection of four spheres or using the Newton-Raphson Algorithm in the case that the solution is non-linear. The tracking of a wireless signal put out by a device could aid in the small scale localization process.

5.2 Summary of the Problem

There are two major discrepancies between tracking a wireless signal and the radio ranging. The first is that the system is flipped, as the signals used for tracking come from a single source being measured at a number of locations rather than measuring several signals from known sources. This does not affect the math behind the system so much as it does the physical design. What greatly alters the necessary computations is the fact the source of the signal is unknown and cannot be directly related to a signal strength. This means that the ranges to any receivers cannot be directly determined from the signal received in a trivial way. Being unable to determine the ranges, a new solution to the localization aspect of the problem must be explored. The problem requires finding the location of the source of the signal using the ratios of its distances from one source to another. This means that is a signal is measured at points A, B, and C, then only the ratios A/B and B/C can be used in the calculation of an analytical solution.

5.3 Solutions to the Problem

5.3.1 Mapping

One potential solution to this problem is to create a map of locations and corresponding signals, then interpolate from that map the source location of the signal we are tracking. The advantage to this solution is that checking whether or not a solution is correct is still very easy for localization. Signal strength decreases at a rate proportional to the inverse square of the distance from the source, so a map could be generated by simple pointwise calculation of the expected signal ratios from a source at a given location. This can be used to produce topographical maps of the signal strength ratios.
that when compared can be used to estimate a source for the signal. The following is an example map with receiver locations at (5, 5), (5, 15), and (15, 10).

![Example map of signal strength at given points.](image)

**Figure 5.1:** Example map of signal strength at given points.

### 5.3.2 Analytical

The two-dimensional problem can also be approached analytically by using the equation of a circle:

\[
R^2 = (X - X_1)^2 + (Y - Y_1)^2 = \frac{S}{S_1^2}
\]  

(5.1)

Where \((X_0, Y_0)\) is the receiver location, \(S\) is the signal strength at 1 meter distance from the receiver, and \(S_1\) is the measured signal strength. To simplify the problem we will place the first receiver at the origin and the second on the x-axis, the third can be located at any x-y coordinate. This is a valid simplification because we can redefine the coordinate frame by translation to center the origin and a rotation to align the second receiver. The final solution can then be rotated and translated back into the original frame.

By solving the equations of the first two receivers for \(S\) and substituting, we arrive at the equation:

\[
\frac{(X^2 + Y^2)S_1^2}{((X - X_2)^2 + Y^2)S_2^2} = 1
\]

(5.2)
This can be solved for:

\[ Y^2 = \frac{S_2^2(X - X_2)^2 - S_1^2X^2}{S_1^2 - S_2^2} \]  

(5.3)

Similarly for receivers 1 and 3 we arrive at the equation:

\[ \frac{(X^2 + Y^2)S_1^2}{((X - X_3)^2 + (Y - Y_3)^2)S_3^2} = 1 \]  

(5.4)

By substituting the values for Y into this equation we get:

\[ \frac{(X^2 + \frac{S_2^2(X - X_2)^2 - S_1^2X^2}{S_1^2 - S_2^2})S_1^2}{((X - X_3)^2 + ((\frac{S_2^2(X - X_2)^2 - S_1^2X^2}{S_1^2 - S_2^2})^{1/2} - Y_3)^2)S_3^2} - 1 = 0 \]  

(5.5)

An analytical solution to this equation does not exist, however, the Newton-Raphson Algorithm can be employed to find a solution for X that satisfies the above equation. The solution is the final X coordinate of the source of the wireless signal. This can be substituted into the original equation which can be then solved for Y. The problem with this method is that the Newton-Raphson Algorithm solves for a zero, but necessarily the zero we are looking for. There is no way of guaranteeing that we ever arrive at the correct zero using Newton-Raphson. This method is therefore unreliable for our purposes.

### 5.3.3 Combination of Analytical and Iterative

The third method for finding the source of a signal uses a combination of an analytical approach and an iterative approach. This approach uses an error function and sampling to approach the true position of the signal source. We will again start with the equation of a circle:

\[ R^2 = (X - X_1)^2 + (Y - Y_1)^2 = \frac{S}{S_1^2} \]  

(5.6)

This time, however, we will essentially try to guess the value of S. This is done by selecting a reasonable range of values for S that place the signal in a certain radius of the first receiver. From here the problem is reduced to a trivial problem of trilateration, calculating the intersection of three circles. Any two circles will have two points of intersection, either real or imaginary, and any three circles will have six intersections between them. Here you can see a configuration with six unique, real intersection points, as well as a configuration with six real intersection points with three in the same location.

![Figure 5.2: Trilateration with six unique intersections.](image)
The error function measures the perimeter of the triangles formed by the intersections of the circles. That is, one of the two intersections from circles one and two, one from circles two and three, and one from circles one and three. This leaves a total of eight possible triangles. From these eight triangles, the midpoint of the triangle with the smallest perimeter is the most accurate estimation of the position of the signal source. The following graph shows the value of the error function for the same receiver configuration shown above, (5, 5), (5, 15), and (15, 10). The recorded signal strengths are (0.0722, 0.4188, 0.1346) at receivers one, two, and three.

![Figure 5.3: Perimeter vs scale factor.](image)

The scale factor is the estimation of the signal strength at one meter, and is clearly at a minimum as a signal strength of 7Wm-2. Using this scale factor and the measured signal strengths we are able to trilaterate the location of the source as (9, 14).

![Figure 5.4: Example of three positions and corresponding signal strengths.](image)
Polling the error function is done in such a way as to focus on tracking objects in a reasonable range, and a reasonable resolution within that range. An initial minimum is estimated from a lower resolution sampling of the entire workspace, followed by a higher resolution sampling from within a margin of error of the estimated minimum from the first sampling. The size of the workspace should be determined based on the range at which you expect to be tracking any signal.

Due to the nature of the problem, there can be multiple positions that will produce the same signal ratios. In the case that there are multiple zeros in the error function, the one with the lowest scale factor is taken to be the solution as it is the position that will be closest to the center of the receiver array, where accuracy will be highest. This is an unavoidable problem when only using three receivers as all zeros to of the error function are valid mathematical and physical solutions for the position of the signal source. A fourth receiver can be used to prevent this issue.

This method has been tested to work in almost all configurations but needs to be slightly adjusted for the case where the receivers are co-linear. In this case every solution will have an equivalent solution mirrored over the line on which the receivers fall. The only solution to this is to assume all objects being tracked are on one side of the receiver array and to ignore the second solution.

Figure 5.5: Example of three positions where there are two solutions.
Sample Results

Figure 5.6: Sample results 1.

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>Scale factor</th>
<th>Actual position</th>
<th>Sensor positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0590</td>
<td>0.0535</td>
<td>0.0118</td>
<td>1.7217</td>
<td>(4,7)</td>
<td>(2,2), (8,3), (13,-1)</td>
</tr>
</tbody>
</table>
Figure 5.7: Sample results 2.

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>Scale factor</th>
<th>Actual position</th>
<th>Sensor positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0270</td>
<td>0.1515</td>
<td>0.0187</td>
<td>7.6081</td>
<td>(18,31)</td>
<td>(13,15), (25,30), (0,40)</td>
</tr>
</tbody>
</table>
### 5.4 Scalability into Three Dimensions

The problem of localization in three dimensions adds different challenges to each solution. In all cases it will require four receivers.

The topographical approach is still valid in three dimensions, however, the analysis and representation become far more difficult. Instead of two topographical maps, there need to be three 3-Dimensional density functions. Instead of comparing two lines of equal potential, we must find the overlap of three surfaces of equal potential. While not impossible, this method is computationally heavy and inelegant.

The analytical approach has the same flaw as the 2-Dimensional version, only compounded. The equation for a sphere is:

\[ R^2 = (X - X_1)^2 + (Y - Y_1)^2 + (Z - Z_1)^2 = \frac{S}{S_1^2} \]  

(5.7)

And the system of equations to solve is:

\[ \frac{(X - X_n)^2 + (Y - Y_n)^2 + (Z - Z_n)^2 S_n^2}{(X - X_{n+1})^2 + (Y - Y_{n+1})^2 + (Z - Z_{n+1})^2 S_{n+1}^2} = 1 \]  

(5.8)

For \( n \) equals 1, 2, 3, and 4.

This mathematical model has even more potential false zeros than the 2-Dimensional model and no analytical solution. It is therefore not plausible to use this model to track a signal.

The combined approach does scale into three dimensions reasonably well. Quadlateration is only slightly more difficult than trilateration, and an analytical solution is easily attainable. Because the error function differs only with the strength of the signal at one meter, it is no more difficult to apply this algorithm to a 3-dimensional model than a 2-dimensional model. Rather than tracking the perimeter of eight possible triangles, we track total length of the edges of sixteen triangular pyramids. Because an analytical solution for each sample is easily attainable, the three dimensional tracking is only marginally more processing heavy than the two dimensional version and is therefore the best solution for extrapolation.
Bibliography


