Summer 2013

Position-adaptive localization of an electromagnetic source using static and mobile wireless sensor networks

Miguel D. Gates

Follow this and additional works at: https://digitalcommons.latech.edu/dissertations

Part of the Other Electrical and Computer Engineering Commons
POSITION-ADAPTIVE LOCALIZATION OF AN ELECTROMAGNETIC SOURCE USING STATIC AND MOBILE WIRELESS SENSOR NETWORKS

by

Miguel D. Gates, B.S., M.S.

A Dissertation Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

August 2013
We hereby recommend that the dissertation prepared under our supervision by Miguel D. Gates entitled Position-Adaptive Localization of an Electromagnetic Source Using Static and Mobile Wireless Sensor Networks be accepted in partial fulfillment of the requirements for the Degree of Ph.D Engineering.

Supervisor of Dissertation Research
Head of Department
College of Engineering and Science

Advisory Committee

Approved:
Director of Graduate Studies
Dean of the Graduate School
Dean of the College
ABSTRACT

We present new results in detection and localization of a hidden emitter using Wireless Sensor Networks (WSNs). While this type of problem has been explored extensively, this research combines multiple dynamics, such as mobility and adaptation, to extend the value beyond the typical search mission. Using Micro-Aerial Vehicles (MAVs), in conjunction with multiple static wireless sensor nodes, we create a hybrid sensor network capable of detection and localization of a hidden electro-magnetic (EM) emitter.

In order to localize the emitter, a Received Signal Strength Indicator (RSSI) is used as an approximation of distance from the transmitter to the revolving receivers. As a result, an algorithm for on-line estimation of the Path Loss Exponent (PLE) is used in modeling the distance based on Received Signal Strength (RSS) measurements. Based on surrounding sensors’ RSS values, the emitter position estimation is then calculated using Least-Square Estimation (LSE) method.

RSS-based localization techniques can be inaccurate due to the noisy and uncertain nature of RSS in different mediums and environments. In order to improve localization accuracy, a technique called Position-Adaptive Direction Finding (PADF) is developed in which a team of MAVs coordinate their sensing missions, adapt their position in real-time, and localize the unknown emitter. We enhance the adaptation segment of the PADF by providing an algorithmic framework for MAVs to reposition
themselves, thus avoiding obstructions or locations that may distort the propagation of the emitter and reduce the accuracy of the receivers’ combined emitter location estimation. Given the cross-PLEs between the static and mobile nodes, we propose a cost function for MAVs’ position adjustments that is based on the combination of such cross-PLEs and RSSIs. The mobile node adjusts current position by minimizing a quadratic cost function such that the PLE of surrounding receivers is decreased, while increasing RSSI from the mobile node to the target, thereby, reducing the inconsistency of the environment created by echo and multi-path disturbances. In the process, the mobile node moves towards a more uniform measuring environment that ultimately increases localization accuracy.

This Dissertation presents our recent results on a novel, multi-platform, RF emitter localization PADF technique. The position-adaptive approach shows potential for an accurate emitter localization in challenging, embedded, multi-path environments such as urban environments. We also present a recent development of a three-state machine-based MAV cooperative control algorithm that is used in search for a hidden emitter.

We provide simulation and experimental results that illustrate proposed methods. We describe the testbed and laboratory development for experimentation and discuss obtained results. Future work is proposed that includes complex cepstrum between mobile nodes and the hidden emitter as a metric for MAV control. Complex cepstrum is correlated with a received echo in EM signals and reducing the cepstrum is expected to improve measurements and estimation accuracy by moving the MAV sensor nodes towards low echo space.
APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author [Signature]
Date 7/22/13
DEDICATION

I dedicate this to my mother—who’s love, strength, and guidance built a strong foundation for me, and to my two sons, Mackenzie and Kelby—I strive each day to be the best I can for you. I hope this makes you as proud of me, as I am of you. I love you.
# TABLE OF CONTENTS

ABSTRACT ............................................................................................................................ ii  
DEDICATION ......................................................................................................................... iv  
LIST OF TABLES ................................................................................................................. ix  
LIST OF FIGURES ............................................................................................................... x  
ACKNOWLEDGMENTS ...................................................................................................... xiii  
CHAPTER 1 INTRODUCTION ................................................................................... 1  
  1.1 Motivation .............................................................................................................. 2  
CHAPTER 2 BACKGROUND ...................................................................................... 5  
  2.1 Wireless Sensor Networks .................................................................................. 5  
  2.2 Localization Techniques using WSNs .............................................................. 6  
  2.3 RF Propagation and Path Loss ......................................................................... 9  
    2.3.1 Sensing Model .......................................................................................... 10  
  2.4 Mobility in WSNs .............................................................................................. 12  
  2.5 Contribution ........................................................................................................... 13  
    2.5.1 Mathematical Models ............................................................................. 14  
    2.5.2 Complexity of Environment .................................................................. 15  
    2.5.3 Mobility ..................................................................................................... 15  
CHAPTER 3 POSITION-ADAPTIVE DIRECTION FINDING ................................. 17  
  3.1 Introduction ........................................................................................................... 17
LIST OF TABLES

Table 4.1: All simulation results of average runtime in search phase. .................. 41

Table 6.1: Sample RSSI values based on varying distance................................. 56

Table 7.1: IRIS platform for multiple configurations with error (case 1-5 corresponds to configurations 1-5 from Figure 7.3 and Figure 7.5) ... 73

Table 7.2: IRIS platform, diamond configuration, with varying distance .......... 74

Table 7.3: Multiple configurations with emitter hidden in a shielded container using weighted-position estimation algorithm................................. 77

Table 7.4: Comparison of weighted algorithm versus non-weighted algorithm .. 78

Table 7.5: Error of position placement using random configurations of static receivers. ................................................................................. 79
LIST OF FIGURES

Figure 3.1: Initial PADF state transitions into next stage after multi-platform adaptation ........................................................................................................ 19

Figure 3.2: Model of a sensor network used to localize a stationary non-cooperative emitter ........................................................................................................ 20

Figure 4.1: Cooperative control of UAV/MAVs for hidden emitter localization .. 28

Figure 4.2: Three-phase diagram for controllers—GS-Global Search, ADT-Approach Detected Target, PADF-Position-Adaptive Direction Finding ........................................................................................................ 28

Figure 4.3: Search map ....................................................................................................... 30

Figure 4.4: MAV movement ............................................................................................... 32

Figure 4.5: Map after partitions and MAV movement ...................................................... 34

Figure 4.6: Search algorithm trajectory with two MAVs. ............................................... 39

Figure 4.7: Runtime of search algorithm with 10 MAVs ............................................... 40

Figure 4.8: Runtime of search algorithm with 5 MAVs ............................................... 41

Figure 5.1: Conceptual Micro-Aerial Vehicle and Sensor Network (MAVSeN) Laboratory ......................................................................................................... 44

Figure 5.2: Fluorescent Tracking Marker—3mm, 9mm, and 14mm markers, from left to right ........................................................................................................ 45

Figure 5.3: Data stream between motion capture cameras and MAVs. ....................... 46

Figure 5.4: Vicon T-40 camera ......................................................................................... 48

Figure 6.1: Adaptive behavior projection ........................................................................ 51

Figure 6.2: Experimental RSSI calibration results for unobstructed emitter ...... 57
Figure 6.3: Emitter inside obstruction during RSS testing........................................ 58
Figure 6.4: RSSI calibration testing with emitter obstructed experimentation results. .................................................................................................................. 59
Figure 6.5: Screenshot of MATLAB simulator for adaptive position algorithm... 61
Figure 6.6: Simulation with single object as a "wall". The simulation was run five times for various values of $R$ and $Q$, thus changing the trajectory/path of the mobile node............................................................. 64
Figure 6.7: Simulation with target encased in "box". The object restricted EM propagation, except to front of box, thus creating a leakage point. The simulation was run five times for various values of $Q$ and $R$...... 65
Figure 7.1: Modified IRIS sensor node for improved omni-directional RSSI measurements................................................................................................... 68
Figure 7.2: Quanser Qball X-4 micro-aerial vehicle integrated with IRIS wireless sensor node. ........................................................................................................ 70
Figure 7.3: Receiving sensor nodes in a diamond configuration surrounding transmitting node. ................................................................. 71
Figure 7.4: An intersection of circles with radii equal to RSS measured at individual nodes; width of disks simulate RSS uncertainty.................. 72
Figure 7.5: Configurations or topologies 2-5. from top left, clockwise. .......... 72
Figure 7.6: A setup for the static experiment with emitter in box obstructing propagation................................................................. 76
Figure 7.7: Testing accuracy of position placement using random configurations of receivers................................................................. 79
Figure 7.8: Experimental position adaptive direction finding (PADF) concept using MAVs with wireless sensor nodes.............................................. 80
Figure 7.9: MAV lab tests: localization error and PADF metric for different experiments................................................................. 81
Figure 7.10: RSSI peak value measurement.......................................................... 82
Figure 7.11: RSSI peak value measurement from behind the box ......................... 83
Figure 7.12: Analytical model for investigating the development of intelligent multi-path echo trend computations (cepstrum). .............................................. 84

Figure 7.13: Cepstrum experimental setup in MAVSeN Lab........................................ 85

Figure 7.14: Cepstrum experiment with trajectory plotted............................................. 85

Figure 7.15: Cepstrum results of experiment without obstruction...................................... 87

Figure 7.16: Cepstrum result of experiment with obstruction.......................................... 87
ACKNOWLEDGMENTS

I would like to thank Dr. Rastko Selmic, my advisor, for not only giving me direction, guidance, and mentoring, but for treating me like a colleague. His help throughout this process cannot be overstated.

I would like to thank the late Dr. Atindra Mitra for serving as a mentor. His brainchild was the foundation upon which this work was based, and his tutelage at the Air Force Research Lab was very valuable. I would also like to thank my other mentors, such as Hector Guevara, Sean Young, and Dr. Raul Ordonez, at the Air Force Research Lab and University of Dayton, for helping me during the many summers I spent interning.

I would like to acknowledge Dr. Vir Phoha, Dr. Travis Atkison, Dr. Jean Gourd, Dr. Jinko Kanno, and Dr. Christian Duncan for serving on my graduate committee and offering wisdom and advice when necessary. In addition to my committee, I must also thank my research group—Saba Ramazani, Chris Barber, Andrew Gardner, Zhang Guyu and Jianfeng Li, who have each helped me in one way or another tackle a problem that I could not solve on my own. Thanks to the many faculty and administrators of Louisiana Tech University whom have aided me on this journey and help make my load lighter and my path clearer.

I would also like to thank my tag-team partner, Justin Rice. We started on this journey together to accomplish this goal, and we grew as students and as friends,
helping each other in any capacity we could. In addition to Justin, my Jackson State board of directors have also helped bridge my time between Jackson State to Tech.

Last but not least, I would like to thank all of family, loved ones, and close friends for their motivation and prayers. And special thanks to Lorrie Harris for being a great support system and backbone. The sacrifices she made to help me get this far did not go unnoticed. Again, thank you all for helping me accomplish what I set out to do.

Through God, all things are possible. Phil. 4:13
CHAPTER 1

INTRODUCTION

Wireless sensor networks (WSNs) are being used in a variety of ways—from search and reconnaissance in military-type exercises [1,2], to biomedical applications [3], to a plethora of commercial endeavors [4–8]. With their low cost, interchangeable sensors, and efficiency, WSNs are very useful in the domain of localization and have become prevalent in missions of detection and localization [9–14]. Using mobile agents have added a new dimension to the problem, as mobile agents have the ability to track a mobile target, actively search an environment, or avoid obstacles that otherwise static agents would have difficulty performing [15–21]. This research explores the use of mobile agents, in the form of Micro Aerial Vehicles (MAVs), in a mission of detection and localization. The MAVs are mobile nodes in 3-dimensional space that have both sensing and communication capabilities. Together with other static sensor nodes, they create a mobile sensing network capable of executing tasks cooperatively.

Aiding in localization, a technique called Position-Adaptive Direction Finding (PADF) was developed, in which a team of MAVs coordinate their sensing missions, adapt their position in real-time, and localize the unknown, hidden Electromagnetic (EM) source based on optimal detection algorithms to acquire better estimation accuracy [22]. In order to localize the emitter, Received Signal Strength Indicators
(RSSI) are used as an approximation of distance from the transmitter to the revolving receivers. As a result, an algorithm for an on-line estimation of the Path-Loss Exponent (PLE) is used in modeling the distance based on Received Signal Strength (RSS) measurements. The emitter position estimation is then calculated based on surrounding sensors’ RSS values using Least-Square Estimation (LSE).

1.1 Motivation

Localization of sensor nodes itself is not a new problem. Many techniques already exist [23–25], and this Dissertation is not an attempt to create a new one; rather, it is an attempt to apply new techniques and technologies to the localization problem. Localization can be very accurate depending on the techniques employed. Range-based radio-frequency (RF) localization methods, which incorporate using techniques such as time of arrival (ToA), time difference of arrival (TDoA), angle of arrival (AoA), and RSS have been explored extensively [11]. While each technique has its advantages and disadvantages, RSS is usually simpler to implement than the others, as it requires no extra hardware or extensive calculations, although some accuracy is sacrificed due to the sensitive and noisy nature of RSS values. For a less accurate (but less computationally and hardware extensive) localization technique of RSS, we improve accuracy by proposing a Position-Adaptive Direction Finding method, in which a sensor swarm adapts their position based on the sensing values and converges towards leakage points in order to detect a hidden EM source. By using the relationship between RSS values and the associated distance between sender and receiver, the transmitter position can be approximated. Research in [26, 27], show
variations of implementing an RSSI-based localization experiment using wireless sensor networks. Specifically, each use a series of reference nodes surrounding a target node when performing localization, while noting that variances in estimation accuracy can be attributed to parameters such as distance, types of obstacles, and density of network. While the research was very thorough in addressing the capability and accuracy of RSSI-based localization, it failed to introduce scenarios that incorporated two factors—mobility and impedance of emitter propagation during location estimation.

Mobility in sensor networks and their applications in sensing, localization, and control have gained significant interest with the development of sensor networks and modern control algorithms \[10,13–15,28\]. Using mobile networks, cooperative control algorithms can be used to maximize the probability of detecting a target \[13,14,18\]. While various platforms can create mobility, researchers have begun to add mobility by interfacing wireless nodes with MAVs or Unmanned Aerial Vehicles (UAVs). Each mobile node is an independent agent capable of distributed sensing and control.

This Dissertation presents a set of experiments using static and mobile sensor networks to localize a cooperative emitter based on measured RSS at surrounding sensor network nodes. Related results are given in \[7,29,30\]. By integrating mobility with sensing functionality, I push range-based localization a step further, as similar research usually consists of purely static nodes. In addition, we also factor in path-loss metrics in our emitter estimation due to environmental changes, such as obstacles, by incorporating a weighted distance estimation of individual receivers using the cross-PLE of surrounding receivers. This is also prevalent in cases when the emitter is shielded or obstructed. Additionally, we use an on-line estimation of the PLE to
model the distance based on measured RSS. A detailed analysis on PLE estimation and modeling is given in [31], which is then used in distance calculations based on RSS measurements. Given that emitter propagation is obstructed during testing, we present an integration of wireless sensor networks and MAVs into one unified localization system and apply adaptive behavior to improve localization accuracy. We explore the effects of multi-path echoes on the propagation of the EM signal and the resulting accuracy on estimation. This integration was tested during a set of experiments performed at Louisiana Tech University’s Micro Aerial Vehicles and Sensor Networks (MAVSeN) laboratory. Also performed in this lab was a set of experiments that formalized using a cepstrum analysis as a tool to identify large objects in the search environment. Knowing the proximal locations of these obstructions allow the adaptive metric to better predict positional movement of the mobile nodes.
CHAPTER 2

BACKGROUND

2.1 Wireless Sensor Networks

A wireless sensor network consists of a collection of sensing devices that can coordinate their actions through wireless communication and aim at performing tasks such as reconnaissance, surveillance, target tracking or environmental monitoring over a specific region. WSNs can be created like a typical ad-hoc networks, but with more dense deployment. Likewise, instead of point-to-point communication between nodes, information can be broadcast to surrounding nodes. Although WSNs are limited in power, computational capacities, and memory, they can be configured with many different types of sensors such as seismic, thermal, visual, infrared, acoustic and radar, which are able to monitor a wide variety of ambient conditions, thus making them suitable for many applications that incorporate continuous sensing, even detection/identification, location sensing, and local control of actuators [32].

These applications range from military applications, to environmental monitoring and commercial applications, i.e., monitoring product quality [33] or environmental control in office buildings [5]. Wireless sensor networks can be an integral part of military command, control, communications, computers, intelligence, surveillance, reconnaissance and tracking (C4ISR) systems. Sensor nodes can be used to monitor
friendly forces, as battlefield surveillance, for reconnaissance of opposing forces and terrain, chemical detection, and targeting. Because WSNs deploy multiple low cost, disposable sensor nodes, destruction of some of these nodes by hostile actions do not deter mission integrity as much as destruction of traditional sensors would [34].

2.2 Localization Techniques using WSNs

In addition to the previously mentioned applications, a prevalent application of WSNs has been localization. Localization refers to the collection of techniques and mechanisms that are used to measure spatial relationships between objects in the physical world by combining raw sensor data with spatial information [32]. There are multiple techniques for localization of RF signals such as Time of Arrival (ToA), Angle of Arrival (AoA), Time Difference of Arrival (TDoA), and Received Signal Strength. There are advantages and disadvantages of each method.

ToA and TDoA methods use geometric relationships based on distances or distance differences between a mobile station and a number of fixed terminals to determine the position coordinates of the mobile target. Data for distance estimations are derived from the arrival times of radio signal epochs at one or more receivers. The ToA method uses the transit time between a transmitter and a receiver directly to find distance, whereas the TDoA method calculates location from the differences of the arrival times measured on pairs of transmission paths between the target and fixed terminals. Both ToA and TDoA are based on the time-of-flight (TOF) principle of distance measurement, where the sensed parameter, time interval, is converted to distance by multiplication by the speed of propagation. In ToA, location estimates
are found by determining the points of intersection of circles or spheres whose centers are located at the fixed stations and the radii are estimated distances to the target. TDoA locates the target at intersections of hyperbolas that are generated with foci at each fixed station of a pair. TDoA-based approaches provide superior ranging results but need more complex and additional hardware, which also adds to energy consumption [35].

The AoA approach to distance measurement and localization is probably the oldest method and easiest to understand in that all that is needed is a directional antenna. AoA methods are the core of direction finding and is not restricted by the problems dictating conditions of use of other location methods. It requires no cooperation from the target, and any type of signal can be used. It also is used over wide frequency bands and ranges—from HF through microwave and from direct true line-of-sight to long communications distances propagated through the ionosphere. AoA is a principle component in a radar system. Using radar, only one fixed station is required to determine the location of a target in two or three dimensions. The two methods of AoA and TOF are employed. When using AoA alone, at least two fixed terminals are required, or two separate measurements by a single terminal in motion [36]. Using AoA to localize hidden targets is not uncommon. In [37], authors used multiple UAVs to cooperatively search for a hidden RF target using a simple AoA sensor to collect target data.

RSS is defined as the measurement of the power present in a received radio signal. Distances are found from signal strength using formulas for propagation. Location coordinates can be calculated based on range estimations acquired using
a propagation model, such as the log-normal shadowing model. The method of finding location from the geometry of the intersection of circles (two dimensions) or spheres (three dimensions) is the same as that used in TOF ranging location systems. Environmental conditions may be accounted for by choosing the propagation law parameters that are most appropriate to the area where the system is used. For most environments, location errors are significantly greater for RSS ranging and geometric location than with TOF methods [36].

An advantage of RSS-based approaches is that these methods are simpler to implement and work with pre-existing hardware, thus cutting costs of additional hardware. There are number of localization algorithms which are based on RSSI. Research involving RSS and localization [26,27,38-41] revolves around using static (or in some cases, mobile) nodes to localize a “hidden” target. Each research differs by RF sensor used, number of receiving nodes deployed, number and mobility of target(s), and position estimation technique. Though different, each has shown a propensity to experience inaccuracy due to the noisy nature of RSS. In order to achieve more accuracy using only RSS as the preferred localization method, additional algorithms have been added to improve existing techniques. In [42], authors proposed a new algorithm, Localization with Ratio-Distance, which compensates when the measurement errors associated with the estimated distance to anchor nodes are high. It also has an added benefit of using the intersection of only two circles, opposed to the standard three of trilateration. Similarly, [39] devised a self-organizing and
practical RSS-based localization technique called EasyLoc, a model-based plug-and-play localization method, that improves on previous approaches in terms of ease of deployment and ease of implementation while still providing a reasonable accuracy.

PADF is similar to these techniques in that it is implemented to enhance the accuracy of the pre-existing RSS method, by compensating for accuracy in localization due to RSS ambiguity by moving to positions of higher accuracy. The technique adds efficiency and accuracy to the RSS method of localization such that it produces comparable results to that of other techniques, like TDoA or AoA, which have higher accuracy, but take considerably more sophisticated hardware to implement.

2.3 RF Propagation and Path Loss

There are specific problems in implementing location awareness with the RSS method. Because of large variations of signal strength due to interference and multi-path on the radio channel, location accuracy is generally less than what can be achieved using TOF methods. Propagation is location/environment specific, and system software usually has to be tailored to the place where the system is being used. Orientation of a target as well as its location related to nearby objects will have an effect on the location estimation.

This phenomena is due in part to loss of an RF signal by many effects, such as free-space loss, refraction, diffraction, reflection, aperture-medium coupling loss, and absorption. This loss can be influenced by terrain contours, environment, propagation medium, the distance between the transmitter and the receiver, and the height and location of antennas.
Radio wave propagation is defined as the transfer of energy by EM radiation at radio frequencies [43]. This energy will attenuate or dissipate as it propagates further from its originating source. The attenuation model is described below.

2.3.1 Sensing Model

I consider a sensing model that is based on the RSS. The RSS gradually decreases with the distance from the transmitter. The RSS is inversely proportional to the square of the distance between the transmitting and receiving antennas. Mathematically, it can be written as

$$\frac{P_r}{P_t} = \left(\frac{\sqrt{G_t} \lambda}{4\pi d}\right)^2,$$

where $P_r$ is received power, $P_t$ is transmitted power, $\lambda$ is the wavelength of the signal, $d$ is the distance between receiver and transmitter, and $G_t$ is the product of the transmit and receive antenna field radiation pattern. Because the antennas on the nodes are omnidirectional, the factor $G_i$ is simply 1. I use this model based on the assumption of free space path loss with receiver and transmitter having line of sight (LOS). Consider $S \subset \mathbb{R}^2$ to be an area where a hidden emitter is located. A network of $N$ equal sensing agents is deployed over $S$ with sensor locations at $(x_i, y_i, z_i)$. A sensing function is given by $p_i(q)$ where $p_i$ is the probability that an event $q \in S$ will be detected. Such probability represents the sensing model that is considered in cooperative MAV control algorithm. I assume that all sensors have equal sensing model and is given by [44]:
\[ p_i(q) = \begin{cases} \frac{R^2 \varepsilon}{d_i^2 \sqrt{R^2 + d_i^2}}, & d_i \leq R \\ 0, & d_i > R, \end{cases} \]  

(2.2)

where \( R \) is the sensing radius of the omnidirectional antenna, \( d_i \) is a distance between the location of a sensor node \( i \) and a specific sensing point in the field \( q \), i.e.

\[ d_i = \| (x_i, y_i, z_i) - q \|, \]  

(2.3)

and \( \varepsilon \) is a small constant that represents the probability of detection at the limit distance \( d_i = R \). Note that \( p_i(0) = 1 \) as \( d_i \) approaches zero and \( p_i(R) = \varepsilon \) as \( d_i \) approaches \( R \). The constant \( R \) depends on noise level, antenna patterns, wavelength, power of transmitter, and transmission medium. For a specific noise level \( P_n \), the sensing radius \( R \) in clear air is given by

\[
P_n \quad \frac{1}{P_t} = \left( \frac{\sqrt{G_i \lambda}}{4\pi R} \right)^2, \]

(2.4)

\[ R^2 = \frac{P_t G_i \lambda}{P_n 16\pi^2}, \]

(2.5)

\[ R = \sqrt{\frac{P_t G_i \lambda}{P_n 16\pi^2}}. \]

(2.6)

In addition to attenuation, propagation can be further attenuated if passing through objects or buildings. Transmission through said objects is dependent on its shielding factor. For most cases, the material, as well as its thickness, determines its shielding factor.
2.4 Mobility in WSNs

Even when incorporating WSNs and using RSS to perform localization, most research involves the use of static nodes. Adding mobility to the network allows for dynamic scenarios. Equipping MAVs with wireless sensor nodes, which serve as the communication and sensing agents, transforms otherwise static receivers into a mobile sensor networks. Other research has used mobile networks to perform tasks; therefore, a foundation in the area of MAVs working cooperatively with these sensor networks already exists [45,46].

Li and Cassandras presented a distributed coverage control scheme for cooperating mobile sensor networks [13]. They developed a gradient based algorithm requiring local information at each sensor and maximizing the joint detection probabilities of random events. Akin to this research, Cortes et al. presented control and coordination algorithms for groups of autonomous vehicle networks performing distributed sensing tasks where each vehicle plays the role of a mobile tunable sensor [10].

Bellingham et al. addresses to problem of cooperative path planning for a fleet of Unmanned Aerial Vehicles (UAVs) in uncertain or adverse environments, by modeling for the probability of UAV loss [47]. Similarly, Richards and How implemented a robust Decentralized Model Predictive Control (DMPC) for a team of cooperating UAVs [28]. Using this DMPC each vehicle plans only for its own actions, but still allows the UAVs to communicate relevant plan data to ensure those decisions are consistent across the team. In a simple case such as collision avoidance, DMPC guaranteed constraint satisfaction and offered significant computation improvement,
compared to an equivalent centralized algorithm, for only a small degradation in performance, such as UAV flight time.

Chandler et al. researched the development of cooperative rendezvous and cooperative target classification agents in a hierarchical distributed control system for unmanned aerospace vehicles [15]. For cooperative target classification he developed templates, followed optimal trajectories, and assigned adjacent vehicles to view at complementary aspect angles; hence, he combined these to maximize the probability of correct target classification over various aspect angles. Singh and Fuller developed a receding-horizon optimal control scheme for autonomous trajectory generation and flight control of an unmanned air vehicle in an urban terrain [48]. Because environments may be dynamic, or the vehicles need to change dynamics mid-fight due to sensor or actuator failure, they proposed a Model Predictive Control (MPC) scheme that navigates a vehicle with nonlinear dynamics through a vector of known way-points to a goal, and manages constraints for missions that will require vehicles with increased autonomy in dangerous situations and with tight maneuvering and operational capability e.g., missions in urban environments.

2.5 Contribution

Localization using RSS-based techniques is a very generic problem. Defining things such as mobility, i.e., the use of MAVs or UAVs, target capabilities, or propagation model help to constrict the generic problem to a specific one. I define a very specific problem in this research: using a single mobile agent, along with multiple static agents, to localize a hidden, stationary target using RSS-based techniques.
Research closely related can be found in [11, 26, 27, 38–41, 49, 50]. In these endeavors, the objective was to use RSS as a method of localization and test the accuracy of the technique employed. Many similarities, such as propagation model or type of wireless networks, can be drawn to make parallels with this research and that of others.

For instance, one of the many similarities is the use of a common propagation model. RSS-based systems are highly-dependent on the propagation model used to infer distance. It can be shown that the estimation performance of RSS-based distance inference decreases considerably as distance between source and receiver increases; therefore, RSS-based localization systems primarily perform well in short range scenarios [51]. One of the most widely used propagation models is the log-normal path-loss model, which is a generalization of the free-space Friis equation [52], where power is allowed to decrease at a rate of \((1/d)^n\), where \(d\) denotes distance. A random variable can be added to account for shadowing (large-scale fading) effects. RSS systems are very sensitive to shadowing and to non-line-of-sight scenarios as signal power decreases considerable, causing large estimation errors. Also, they are very dependent on the specific scenario (indoors, outdoors, heavy clutter, etc.), frequency, weather, etc.

2.5.1 Mathematical Models

In [38], Li similarly uses the log-normal path-loss model to localize a hidden emitter using wireless networks. Using collaborative localization techniques such as maximum-likelihood estimator (MLE) and multidimensional scaling (MDS), they attempt to showcase the use of these techniques in complex application scenarios.
with the mindset of improving the performance of traditional localization techniques. While they did show improvement in their testing, my research consists of using a more simplistic mathematical estimator than MLE and MDS, thus making hardware computation easier. I feel this will improve real-time calculation in estimation speed. I can use the PADF method to increase accuracy that is sacrificed for speed.

2.5.2 Complexity of Environment

In [26] and [27], Daiya et al., and Palazon et al., respectively, use a very similar setup to our research, including down to the IRIS wireless sensor network nodes (with omnidirectional antennas) and LSE for localization (Daiya uses an additional Adaptive n-Triangle algorithm). Performing basic localization examples using multiple static receivers and a stationary target, they composed results based on the accuracy of the experiment—estimated position versus actual position. While their algorithms performed well as expected, their analysis was focused on the density and distribution of the receivers. Because their scenarios involved no interference of the source, I expanded this in incorporate a complex scenario by adding obstacles and shielding the emitter in the search environment. Within this complex environment, I compensate for expected path loss by weighting our distance estimation algorithm.

2.5.3 Mobility

In [11], Moragregra studies the effects of mesh and clustered topologies. Similarly, I conduct experiments on configurations that perform better than others as well as how clustering of receivers affect overall performance. Moragregra showed that while communication can be improved among nodes in clusters, localization accuracy
usually suffers. To avoid this setback, I employ the use of mobility to take nodes from areas of lower accuracy to higher accuracy based on movement metrics associated with the PADF concept that is explained in Chapter 6.

In comparison to past research, the contributions of this Dissertation include the following: a modified RSS-based localization technique used to enhance the accuracy of an RSS-based localization system; a simplified location estimation algorithm to improve real-time speed for simple RSSI-distance calculation; a position-adaptive MAV control algorithm for MAV repositioning; development of a simulator that allows multiple cooperative mobile nodes to localize a static target(s) more effectively by moving to areas of higher localization accuracy; and a laboratory test-bed for real-time testing using MAV platforms. In essence, I propose to use MAVs to create a hybrid sensor network that is capable of localizing a cooperative, shielded emitter based on measured RSS at surrounding sensor network nodes. Active localization techniques emit signals into the environment that are used to measure range to the target. The emitter is considered cooperative because it emits a signal with known characteristics, and elements of the system (receiving nodes) detect the signals and use information about the signal to deduce the target's location. The PADF method will be used as a supplement to the RSS localization scheme, as the objective is to move from areas of low accuracy into areas of potential higher accuracy.
CHAPTER 3

POSITION-ADAPTIVE DIRECTION FINDING

3.1 Introduction

The concept of Position Adaptive Direction Finding (PADF) was introduced in [22]. PADF is based on the formulation and investigation of path-loss based metrics that are measured and estimated across multiple platforms in order to intelligently "position-adapt" the location of each platform. In other words, a sensor swarm adapts its position based on sensing values, dependent upon the medium, and converges towards leakage points in order to detect a hidden EM source. Typical direction finding is defined as a technique in which an emitter is localized in an open environment, usually using a well-defined method such as AoA, TDoA, or a hybrid of multiple techniques. PADF modified these concepts to encompass localizing an emitter in an urban or embedded environment. Given multi-path and obstacles, the objective is to localize a hidden, cooperative EM signal. Given \( n \) mobile sensor nodes or MAVs, we developed cooperative control algorithms that will detect and localize the EM source. By using the relationship between RSS values and the associated distance between sender and receiver, the transmitter position can be approximated. The Received Signal Strength Indicator (RSSI) from the receiver node’s sensor data is acquired, converted into approximate distance values based on approximated Path
Loss Exponent (PLE) values, and used to estimate the position of the emitter using the Least Square Estimation (LSE) method. The algorithm is based on the LSE method described in detail in [4].

Mobile sensor networks and their application in sensing, localization, and control have gained significant interest with the development of sensor networks and modern control algorithms [53, 54]. Cooperative control algorithms that maximize the probability of detection are given in [13, 14]. In addition, control and coordination for groups of autonomous vehicles performing distributed sensing was presented in [10]. With the development of MAVs, cooperative control and sensing gained attention for applications such as military, weather forecast, chemical sensing, etc. A problem of cooperative path planning for a fleet of unmanned aerial vehicles in uncertain environments was presented in [15]. Similarly, a robust decentralized model predictive control for a team of aerial vehicles is given in [28] and for situations in urban terrain or urban battlefield in [48].

We present a set of experiments using static and mobile sensor networks to localize a cooperative sensor node based on measured RSS at surrounding sensor network. Related results are given in [29, 30, 55]. I use on-line estimation of the PLE to model the distance based on measured RSS. A detailed analysis on PLE estimation and modeling is given in [31] which is then used in distance calculations based on RSS measurements.
3.2 PADF Formulation

Figure 3.1 illustrates two states of a notional PADF geometry using three UAV’s that are integrated with sensor motes programmed to function as RF receivers. A theoretical signal analysis for this geometry is provided in [57] for purposes of formulating and illustrating this technique. The formulation and investigation of path-loss based (i.e. PLE) metrics that are measured across multiple platforms are intrinsic to this technique.

![Diagram of Initial MAV positions and MAV positions after transition based on PADF](image)

Figure 3.1: Initial PADF state transitions into next stage after multi-platform adaptation.

Given two sensor nodes at locations \((x_i, y_i), (x_j, y_j)\), and an emitter \(e = (x, y)\) in the \(x-y\) plane, the Euclidian distances between the emitter \(e\) and the sensors are given by

\[
\begin{align*}
    r_i(e) &= \sqrt{(x-x_i)^2 + (y-y_i)^2} \\
    r_j(e) &= \sqrt{(x-x_j)^2 + (y-y_j)^2}.
\end{align*}
\]

(3.1)

I assume that the sensors provide enough information, using RSSI measurements, such that \(r_i\) and \(r_j\) can be calculated based on the RSSI data. The problem considered is to determine the location of the emitter \(e = (x, y)\), or equivalently, to solve two
equations for two unknowns in Equation 3.1. The solution to the system of equations is the intersections of the corresponding circles, or radii $r_i$ and $r_j$. However, three receiving nodes are required to precisely determine the position of the emitter.

In case of $N$ surrounding sensors, the problem is to determine the location of the emitter $e$ or to solve the following system of equations

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}, i = 1, 2, ..., N. \quad (3.2)$$

Since sensor measurements are noisy and the system of equations in (3.2) has a larger number of equations than unknowns, I use a LSE to determine the location estimate $(\hat{x}, \hat{y})$.

The PLE depends on the medium of EM signal propagation and is not known a priori. Consider the model given in Figure 3.2 consisting of a network of sensors $S_i \ (i = 1, 2, 3, 4)$ and a cooperative emitter $S_0$. Distance between nodes $i$ and $j$ is denoted as $d_{ij}$. The signal strength received from node $i$ at node $j$ is denoted as $P_{ij}$.

![Figure 3.2: Model of a sensor network used to localize a stationary non-cooperative emitter.](image-url)
Assumption 1. A symmetric wireless signal propagation model is considered, i.e., $P_{ij} = P_{ji}$, $i \neq j$, and $i, j \in \{0, 1, 2, \ldots, N\}$.

Assumption 2. The location of sensor nodes $S_i$ ($i = 1, 2, \ldots, N$) is known, the location of the emitter $S_0$ is unknown, and $N$ is the number of sensors used in localization. The RSS model that relates RSS and distance is the log-normal shadowing model, and is given by [31],

$$P_{ij}(dBm) = P_0(dBm) - 10 \cdot \alpha \cdot \log\left(\frac{d_{ij}}{d_0}\right), \quad (3.3)$$

where $\alpha$ is the PLE, $P_{ij}(dBm)$ is the power received at node $j$ from node $i$ in dB milliwatts, and $P_0(dBm)$ is a reference power received at some known distance $d_0$, that is measured a priori. Note that the calibration method requires measurements of $P_0$ and $d_0$.

### 3.3 Path Loss Exponent Estimation

The PLE estimation, $\hat{\alpha}$, is given by the following optimization problem, where it is required to find a minimum of the quadratic cost function

$$f(\hat{\alpha}) = \sum_{i,j=1}^{N} (P_{ij}(dBm) - P_0(dBm) + 10 \cdot \hat{\alpha} \cdot \log\left(\frac{d_{ij}}{d_0}\right))^2. \quad (3.4)$$

The PLE estimate is given by

$$\frac{\partial f(\hat{\alpha})}{\partial \hat{\alpha}} = 2 \sum_{i,j=1}^{N} 10 \log\left(\frac{d_{ij}}{d_0}\right) (P_{ij}(dBm) - P_0(dBm) + 10 \cdot \hat{\alpha} \cdot \log\left(\frac{d_{ij}}{d_0}\right)). \quad (3.5)$$

Therefore, the solution is given by

$$\hat{\alpha} = \frac{\sum_{i,j=1}^{N} \log\left(\frac{d_{ij}}{d_0}\right) (P_0(dBm) - P_{ij}(dBm))}{10 \sum_{i,j=1}^{N} \log^2\left(\frac{d_{ij}}{d_0}\right)}. \quad (3.6)$$
Such obtained value of the PLE represents a maximum likelihood estimator value. Note also that calculation of the PLE given by Equation 3.6 requires prior calibration between sensors $S_i$. In these experiments, I use a similar setup as shown in Figure 3.2.

### 3.3.1 RSSI Model on IRIS Sensor Nodes

If distances $r_i$ and $r_j$ in Equation 3.1 are not known, they can be calculated based on RSSI data. Based on MEMSIC’s IRIS wireless sensor nodes [56], the RSSI model used is given by

$$RSSI_{ij} + K = (RSSI_0 + K) - 10 \cdot \alpha \cdot \log(D),$$  

(3.7)

where $\alpha$ is the PLE, $D$ is the distance between transmitter and receiver, $RSSI_{ij}$ is the RSS at a reference distance of one meter, and $K$ is the conversion constant of RSSI to dBm. RSSI itself is a number between 0 and 28 indicating the received signal strength as a linear curve on a logarithmic input power scale (dBm) with a resolution of 3 dB. An RSSI value of 0 indicates an RF input power of $<-91$ dBm; a value of 28 a power of $\geq -10$ dBm [58]. This can be calculated into RF input power by $P_{RF} = RSSI\_BASE\_VALUE + 3 \cdot (RSSI - 1)$.

Therefore, for IRIS sensor nodes, the conversion constant is approximately -91, thus giving the power in dBm of $RSSI_{ij}$ and $RSSI_0$ as $P_{ij}$ and $P_0$, respectively. The distance is can be calculated by

$$D = 10^{-\frac{(P_{ij})-(P_0)}{10 \alpha}}.$$  

(3.8)
3.4 Emitter Location Estimation

Let \( \hat{\mathbf{e}} = (\hat{x}, \hat{y}) \) be the estimated position of the emitter. The matrix of receivers' geometry (known \( x,y \) coordinates of receivers) is

\[
H = \begin{bmatrix}
    h_{x(2,1)} & h_{y(2,1)} \\
    \vdots & \vdots \\
    h_{x(N,1)} & h_{y(N,1)} \\
    \vdots & \vdots \\
    h_{x(N,N-1)} & h_{y(N,N-1)}
\end{bmatrix}_{(N/2) \times 2}
\]

(3.9)

where \( N \) is the number of receivers, \( h_{x(i,j)} = 2(x_j - x_i) \), and \( h_{y(i,j)} = 2(y_j - y_i) \). The matrix of RSSI measurements is

\[
C = \begin{bmatrix}
    c_{2,1} \\
    \vdots \\
    c_{N,1} \\
    \vdots \\
    c_{N,N-1}
\end{bmatrix}_{(N/2) \times 2}
\]

(3.10)

where \( c_{i,j} = r_i^2 - r_j^2 + x_j^2 - x_i^2 + y_j^2 - y_i^2 \). Then, the estimated transmitter location is calculated based on \( H \cdot \hat{\mathbf{e}}^T = C \) and is given by [14]

\[
\hat{\mathbf{e}} = (H^TH)^{-1}H^TC.
\]

(3.11)

3.4.1 Weighted Position Estimation

In environments that have multiple EM obstructions or the emitter is shielded, such that EM propagation is attenuated, the location estimation can be inaccurate. As a receiver moves further away from a target, the resulting RSSI will decrease. A
smaller RSSI produces a larger distance from the emitter. Similarly, an emitter that is shielded will yield lower RSSI than an unshielded emitter at the same distance, thus making the receiver perceive it is further away from the emitter than it actually is. To compensate for EM loss due to shielding and multi-path, the following modified least squares approach can be adopted for this PADF investigation where the distance terms in the $H$-matrix of Equation 3.11 are weighted by the inverse of vector Cross PLE terms that are estimated from a set of cooperative RSSI measurements between receivers. For example, for a four-receiver scenario

$$\overrightarrow{CPLÉ} = [PLE_{1,2} PLE_{1,3} PLE_{1,4} PLE_{2,3} PLE_{2,4} PLE_{3,4}]$$  \hspace{1cm} (3.12)$$

$$MPLE = mean(\overrightarrow{CPLÉ})$$  \hspace{1cm} (3.13)$$

$$\overrightarrow{W} = \left[ \frac{MPLE}{PLE_{1,2}} \frac{MPLE}{PLE_{1,3}} \frac{MPLE}{PLE_{1,4}} \frac{MPLE}{PLE_{2,3}} \frac{MPLE}{PLE_{2,4}} \frac{MPLE}{PLE_{3,4}} \right]$$  \hspace{1cm} (3.14)$$

In a general sense, the vector $\overrightarrow{W}$ can be mapped as

$$\overrightarrow{W}_k = P_{ij}, i = 1, \ldots, N - 1; j = i + 1, \ldots, N$$  \hspace{1cm} (3.15)$$

where $k$ is $(i - 1)(N - \frac{i}{2}) + j - i$, $N$ is the number of receivers. As illustrated by the following derivation, this set of cooperative inverse PLE weights have the relative effect of de-emphasizing measurements that are affected by non-line-of-sight propagation losses due multi-path and materials attenuation. The weighted estimation is then

$$H_W(i, 1) = H(i, 1) \times \overrightarrow{W}_i$$  \hspace{1cm} (3.16)$$

$$H_W(i, 2) = H(i, 2) \times \overrightarrow{W}_i$$  \hspace{1cm} (3.17)$$

$$\hat{e}^* = (H_W^T H_W)^{-1} H_W^T C.$$  \hspace{1cm} (3.18)$$
CHAPTER 4

THREE-PHASE MACHINE FOR SEARCH AND LOCALIZATION

4.1 Introduction

Looking to extend the problem beyond a small environment in which target detection is a near certainty, we broadened the area to approach the problem as if the target was not initially detected. Therefore, a search algorithm was developed to search a large area cooperatively with multiple mobile agents. Cooperative control is a concept in which a group of spatially distributed controlled objects function together for a common objective. Researchers have applied this definition to the use of multiple UAVs or Unmanned Ground Vehicles (UGVs) to cooperate together to complete a common goal more efficiently. Cooperative control among robotic agents has been around for the last 20 years. Among its many applications are recognizance, surveillance, and search in areas of uncertainty or difficulty [17,59]. Expanding cooperative control paradigms to UAVs opened new applications, including to the military, where UAVs are still being used for exploration, but have also been used for attack missions [15]. Recently, cooperative control of UAVs has created interest in the research community to develop more applications. For instance, the research in [60] uses a team of vehicles with consensus algorithms (shared variable of interest),
under which each vehicle responds to the reference state received by the leader while maintaining their formation, similar to a flock of birds. The leader had access to a reference model, which allowed the followers to exchange information through a topology, thus forming a centralized control system.

Typically, cooperative control methods invoke a decentralized or distributed control system of UAVs and mobile robotic units, as in [9,15,17,45,46,53,61]. Similarly, in [46], UAVs are cooperatively controlled via a leader-follower dynamic, in which the leader tracks a planned path while other vehicles maintain platoon formation and follow the leader based on a global reference frame, using a linear time-invariant feedback system. UAVs that are able to rendezvous at a particular location and cooperatively classify targets are presented in [15], where the flight trajectories were constructed using the Voronoi diagrams approach. A switching algorithm that allows for long-range interactions between with local neighbors was presented in [61]. UAVs with a sense of learning or intelligence via a feedback controller were presented in [45,53].

Most of the research mentioned above allowed communication among the multiple agents, but lacked the ability to sense features about the given search environment; providing sensing capability through WSNs enabled the advent of mobile sensor networks. Mobile sensor networks and their applications in sensing, localization, and control have recently gained significant interest. Research in the area of cooperative control of mobile sensor networks is presented in [10,13,16,19]. An interesting method of cooperative control of multiple UAVs was to use a state machine control architecture to search for, detect, and localize RF emitters [8].
I propose a three-phase machine for a cooperative control of MAVs, equipped with wireless sensor nodes, capable of sensing electromagnetic emissions within a short range in order to localize a hidden EM source. The MAVs are cooperatively controlled based on aforementioned three-phase machine algorithm that implements a decentralized search, optimal control-based navigation protocol, and PADF concept. The PADF phase is used to reduce a localization error by adapting the positions of the MAVs in a real-time based on cross-path loss exponents estimates between MAVs. This concept of PADF has been explored in depth regarding analysis and trends of the localization algorithm, as well as node configuration stability [30,57].

### 4.2 Cooperative Control Phase Machine

Based off the sensing model in Section 2.3.1, I designed a three-phase machine-based control algorithm that controls the localization mission for each MAV. These three phases are as follows:

- Global Search
- Approach Detected Target
- Position-Adaptive Direction Finding.

The first phase, Global Search, controls MAVs in their searching of environment for the pre-specified level of RSS, and once found, MAVs will gravitate towards that particular area. The second phase determines the proper path to take and coordinates multiple MAVs towards vicinity of the hidden emitter. The cooperative control is based on a cost function that moves MAVs to a pre-defined optimal distance from the hidden emitter. The third phase is a PADF where mobile nodes adapt their positions
in order to minimize the metric that is based on cross-path loss exponent estimations (correlated with environment surrounding the hidden emitter). The system and phase diagrams are pictured below in Figures 4.1 and 4.2, respectively. The basis of this system was derived from a similar system in [9]. In this system, $U_i$ represents the pertinent information communicated from other MAVs, while $v_i$ is the MAV's own information.

![Guidance Controller and UAV movement diagram](image1)

**Figure 4.1**: Cooperative control of UAV/MAVs for hidden emitter localization.

![Three-state machine diagram](image2)

**Figure 4.2**: Three-phase diagram for controllers—GS-Global Search, ADT-Approach Detected Target, PADF-Position-Adaptive Direction Finding.
4.2.1 Global Search

The Global Search phase encompasses searching the designated area for the target. Each mobile node is tasked with seeking areas of high signal strength. The method at which this is done is the basis of the search phase. The algorithms to be implemented depend on the information available, such as a global map. For example, if the system is centralized, then each mobile node has full access to a centralized controller or base station, which will allow information to be passed to all nodes simultaneously. This can be beneficial in assigning search areas for individual nodes. However, if the system is distributed, then each node works independently, thus increasing the difficulty of defining an efficient manner of searching an area. Given the different degrees of complexity associated with each system, I make the following assumptions:

Assumption 3: A global map is given \textit{a priori}.

Assumption 4: A centralized control system is used.

Based on the first assumption of the global map given beforehand, the environment is defined below. The goal of the first stage in the three-phase machine algorithm is to address cooperative search in a given environment by a team of MAVs. Although the environment is unknown in regards to target location, there are some assumptions that are made within the search paradigm. This is beneficial when developing a map, and partitioning this map into separate entities of smaller, equal sizes that will be known as cells. The environment will be defined as a rectangular matrix with dimensions $N \times M$, in which $N$ is the number of cells in the x-direction and $M$ is the number of cells in the y-direction. The cells are formed by dividing the matrix into
equal. square parts. Each individual cell's length is based on the sensing radius of the wireless sensor node, $r_s$. Because each node has a circular sensing pattern (due to the omnidirectionality of the antenna), each cell's length is actually $r_s\sqrt{2}$; thus, the area of each cells is $2r_s^2$. The total number of cells given by $N \times M$ is known \textit{a priori} by the first assumption. The concept is illustrated in Figure 4.3.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{search_map.png}
\caption{Search map.}
\end{figure}

\textbf{Assumption 5}: There is only a single stationary target hidden in the environment. Because the environment is of a known size, the objective of the search algorithm is to search through the area of interest and approximately locate an emitter (the next two phases will refine the localization information). This allows MAVs to plan their trajectories towards finding the target versus exploring the environment. Each cell is given a coordinate location $(x, y)$. The target is hidden in one of the cells, but this location is initially unknown.
**Assumption 6:** All cells have equal initial target probability.

I use a sweep search algorithm. This model defines the search region, and gives each MAV a cost function for searching particular areas. The mobile nodes are expected to search in areas of high probability to detect the emitter. In the present case, the higher RSS leads to a higher the probability of detection. The mobile nodes will gravitate towards areas with high RSS and will continue to move in that direction until in appropriate range to localize the emitter. Assuming a uniform probability of a target distribution, based on the fourth assumption, each cell has the same likelihood that a target is present at that specific cell. A very similar approach was done in [45], albeit, in a more complex situation to utilize a probability detection map. If the probability of the target being detected in the space $N \times M$ is 1, then the probability that the target is residing in any specific cell is

$$P = \frac{1}{N \times M}.$$  \hfill (4.1)

As time progresses and cells are searched, the probability is changed accordingly. Therefore, the probability map $P_{(x,y)}(t)$ is a function of time.

Each cell is given two states, $s_{x,y}$ and $t_{x,y}$, in which it is assigned a binary value of 0 or 1 depending on if the cell has been searched, and for a target being present, respectively. For example, if a search occurred in cell (1, 3) and a target was found, then $s_{1,3} = 1$ and $t_{1,3} = 1$; otherwise, $s_{1,3} = 1$ still, but $t_{1,3} = 0$. As the MAVs traverse the environment in search of the target, the states of each cell are taken into consideration and implemented into the updated probability map. The probability of
the target present in each cell is given by

\[ P_{x,y}(t) = \frac{1}{(N \times M) - \sum_{i,j=1}^{N,M} s_{i,j}}. \]  

(4.2)

The MAV will have the ability to traverse from cell to cell in the environment, particularly from the middle of one cell to the middle of the next, as shown in Figure 4.4. The dark figure in the middle cell is the current position of the MAV. The corresponding gray figures are the expected positions that the MAV could move to. Notice, that the MAV can only move into the middle of the adjacent cells, therefore, each one can move in only one of four directions—north, east, south, and west, shown in Figure 4.4 as the red arrows numbered 0-3. These movements are defined as \( m_i(t) = \{0, 1, 2, 3\} \), where 0 corresponds to north, 1 to east, 2 to south, and 3 to west.

Figure 4.4: MAV movement.
Given these assumptions, the algorithm implemented was similar to a sweep search. In accordance to the phase diagram, once a node has reached a level above the threshold, that node will enter into the next phase, while prompting other nodes to converge towards it.

4.2.2 Cooperative Search Algorithm

In a sweep search, the agents will search each cell one at a time in a coordinated fashion until the target is found. Because the global map was supplied as part of our assumption, I can partition the map into smaller, more defined areas for the individual nodes to explore. Based on the number of nodes, $n$, the map will be divided into $n$ subsections. Each node will be assigned to its own subsection, in which it will then perform a sweep search. The search area was previously defined as $N \times M$, where $M$ cells can be divided into $n$ partitions of size $C_i$, where $i = 1, \ldots, n$. In cases of unequal partitions, a remainder, $R$, can be found using $M \% n$ ($\%$ is mod operator). This remainder will be evenly distributed among the available partitions.

Once the environment has been divided into partitions, the nodes will be sorted according to their $y$-position. The higher position will be assigned to the top most cell block, the next highest to the next tier cell block, and so forth, as shown in Figure 4.5. Once assigned to a cell block, the MAV will do a sweep search of that specific block based on pre-defined navigation rules. MAVs will travel to cells with higher values of the cost function $W_{(x,y)}(t)$, where the cost function is defined by $W_{x,y}(t) = P_{x,y}(t) + S_{x,y}(t)$, where $S_{x,y}(t) = s_{i+1,j} + s_{i-1,j} + s_{i,j-1} + s_{i,j+1}$. This states that all cells surrounding a node that have not been searched have a higher probability
of target presence than cells that have. In addition, there is a boundary condition that defines the probability of a cell outside of the designated search area contains the target is zero, thus, ensuring that nodes will choose not to venture to those cells.

![Diagram](image)

Figure 4.5: Map after partitions and MAV movement.

### 4.2.3 Approach Detected Target

Approach Detected Target is the second phase of the phase machine. During this phase, a target has been detected within a search radius, but not localized. Given $N$ mobile nodes, the objective of this phase is for the nodes to navigate towards the target while maintaining a proximal distance from each other. This is accomplished
using by creating an appropriate cost function and using a steepest descent algorithm. Consider a focus point \((X_t, Y_t)\), and a set of \(M\) mobile nodes. The mobile nodes need to converge to the focus point. Given the area of convergence as \(A\). I have to find the optimal vertex location for the \(M\)-mobile nodes such that the following cost function is minimized

\[
\min J_i = R \sum_{i \in M} \{\text{dist}[(x_i, y_i), (X_t, Y_t)] - R_E\}^2 + P \sum_{i, k \in M, i \neq k} \text{dist}^{-2}[(x_i, y_i), (x_k, y_k)],
\]

where \(R\) and \(P\) are the design parameters. The parameter \(R\) controls the closeness to the focus point, while parameter \(P\) controls the distribution of mobile nodes among themselves so as to get the optimal deployment. For example, if \(R >> P\) implies that the mobile nodes should be deployed so as to concentrate on the focus point (target), keeping the mobile node deployment optimal. The factor \(R_E\) represents the optimal distance away from the target that the MAVs should be in order to get the most accurate localization. During previous experiments, it was discovered that \(R_E\) is approximately seven meters for the type of experiments conducted [29].

**Gradient Descent Algorithm for M Mobile Sensor Nodes Deployment**

**Problem:** Find the optimal vertex location \((x_i, y_i)\) where \(i \in M\) such that the following cost function is minimized.

**Step 0:**

\[
\min J_1 = R \sum_{i \in M} \{\text{dist}[(x_i, y_i), (X_t, Y_t)] - R_E\}^2 + P \sum_{i, k \in M, i \neq k} \text{dist}^{-2}[(x_i, y_i), (x_k, y_k)],
\]
As long as $i$ does not equal to $k$, then the equation is differentiable.

**Step 1:**

Calculate

$$\frac{\partial J_1}{\partial x_1} = R \sum_{i \in M} 2(x_i - X_i) - P \sum_{i,k \in M, i \neq k} \frac{2(x_i - x_k)}{[(x_i - x_k)^2 + (y_i - y_k)^2]^2}, \quad (4.4)$$

$$\frac{\partial J_1}{\partial y_1} = R \sum_{i \in M} 2(y_i - Y_i) - P \sum_{i,k \in M, i \neq k} \frac{2(y_i - y_k)}{[(x_i - x_k)^2 + (y_i - y_k)^2]^2}. \quad (4.5)$$

**Step 2:**

Let the start point be $(x_{10}, y_{10})$ and let the tolerance be given by $\varepsilon$.

**Step 3:**

Choose step size as $\lambda_s$ to minimize $J_1$

$$J_1 \left(x_{is} - \lambda_s \frac{\partial J_1}{\partial x_{is}}, y_{is} - \lambda_s \frac{\partial J_1}{\partial y_{is}}\right) = \min J_1 \left(x_{is} - \lambda \frac{\partial J_1}{\partial x_{is}}, y_{is} - \lambda \frac{\partial J_1}{\partial y_{is}}\right), \quad (4.6)$$

where $i \in M$. This step is to be repeated for every mobile node.

**Step 4:**

Calculate the next point $(x_{i(s+1)}, y_{i(s+1)})$

$$x_{i(s+1)} = x_{is} - \lambda_s \frac{\partial J_1}{\partial x_{is}}, \quad (4.7)$$

$$y_{i(s+1)} = y_{is} - \lambda_s \frac{\partial J_1}{\partial y_{is}}, \quad (4.8)$$

where $i \in M$. This step is to be repeated for every mobile node. If the algorithm tries to move two mobile nodes to the same location (unlikely), the request is denied and the step is re-updated.
Step 5:

Terminate the calculation if

$$|J_1(x_{is} - y_{is}) - J_1(x_{(s+1)}, y_{(s+1)})| \leq \varepsilon.$$  \hspace{1cm} (4.9)

Otherwise, repeat Step 4.

4.2.4 PADF

PADF, as described in Chapter 3, is based on the formulation and investigation of path-loss based metrics that are measured and estimated across multiple platforms in order to intelligently position-adapt the location of each platform. In other words, each individual node adapts its position based on sensing values, dependent upon the medium, and converges towards the target. By using the relationship between RSS values and the associated distance between sender and receiver, the target’s position can be approximated. RSS data is approximated into distance values based on PLE values, and estimates the position of the target using a Least Square Estimation method. The convergence and adaptation of the MAVs’ position is the key to the PADF concept. The objective is to reduce the error associated with the localization of the target as the MAVs organize themselves in certain configurations.

4.3 Simulation and Implementation

The control algorithm was implemented in a custom JAVA simulator. This simulator is primarily used to simulate a distributed network with mobile agents and multiple static nodes in search of a target. The static nodes represent an ad-hoc WSN with communication to mobile sensor nodes/MAVs. The simulator implements the
first phase of the three-phase machine MAV control algorithm that provides a search for a hidden electromagnetic emitter. Some of the key components implemented in the simulator include the following:

**Create a search map:** Each node has a stored map of the searched area.

**Define position** \((x, y)\): The position of the mobile MAV is \((x, y)\).

**Navigation laws:** These laws are based on the three-phase machine and how a specific MAV would traverse the environment. Given the search algorithm, boundary constraints and path planning are implemented.

### 4.3.1 Simulation

The simulator was designed to implement the cooperative sweep search algorithm mentioned above. The search area (size of which is known) was partitioned into equal sized sub-cells. Each MAV is then assigned to a specific sub-cell based on the \(y\)-coordinated position. A random target is created in the global search area, and the task of each MAV is to search its predefined area and report back to the other MAVs whether or not it finds the target in a centralized fashion. As seen in Figure 4.6, given two MAVs, a simple sweep search is implemented. The blue and red dashed line would be the expected trajectory of each \(MAV_1\) and \(MAV_2\), respectively. The simulation stops and moves to next stage whenever the target is found by any single MAV, thus not allowing \(MAV_2\) to completely search its given area.

In the current scenario setup, I ran multiple instances of the simulator with varying factors such as size of search area and number of MAVs. The total number
of MAVs range from 1, 3, 5, or 10, while the area is $100 \times 200$ units. I then double this space and rerun all experiments (search area of $200 \times 400$ units). Looking at the case of 10 MAVs working in tandem to search an area that is $100 \times 200$ units, and given the dynamics of the algorithm, this area was divided into equally partitioned sub-cells of $100 \times 20$ units. Since the MAVs are initially randomly positioned, their $y$-coordinate position was sorted, and according to an ID, assigned to a particular cell.

The sweep search was conducted until one of the MAVs searched the area of the target's location. Once this happened, the search is over and the next stage is executed. During the search stage, the results of the algorithm efficiency are shown below. Each MAV moves exactly one unit per step in the search. A total sum of the unit steps represents the runtime of the algorithm. Since the process is running
as a thread, versus as a parallel algorithm, this was the most reliable indicator of a true runtime. I ran the simulation 50 times and plotted the average runtime, as can be seen in Figure 4.7 for the aforementioned case. Figure 4.8 shows the runtime of the same scenario with five MAVs. The rest of the simulation results can be seen in Table 4.1. Notice that as the number of MAVs increases, the runtime decreases. This is to be expected, in that the algorithm is more efficient with more MAVs searching cooperatively.

![Graph showing runtime of search algorithm with 10 MAVs.](image)

Figure 4.7: Runtime of search algorithm with 10 MAVs.
Figure 4.8: Runtime of search algorithm with 5 MAVs.

Table 4.1: All simulation results of average runtime in search phase.

<table>
<thead>
<tr>
<th>Number of MAVs</th>
<th>Size of Search Area (units)</th>
<th>Simulation Count</th>
<th>Avg. Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>100 × 200</td>
<td>50</td>
<td>8321</td>
</tr>
<tr>
<td>5</td>
<td>100 × 200</td>
<td>50</td>
<td>4674</td>
</tr>
<tr>
<td>10</td>
<td>100 × 200</td>
<td>50</td>
<td>1647</td>
</tr>
<tr>
<td>3</td>
<td>200 × 400</td>
<td>50</td>
<td>28520</td>
</tr>
<tr>
<td>5</td>
<td>200 × 400</td>
<td>50</td>
<td>25921</td>
</tr>
<tr>
<td>10</td>
<td>200 × 400</td>
<td>50</td>
<td>15025</td>
</tr>
</tbody>
</table>

4.4 Conclusion

This paper has presented a guide to a phase-machine approach in cooperative control as it relates to a mobile sensor network of MAVs searching for a hidden RF emitter. The results show that with modifications and improvements, this can be viable as a fully implementable search method for UAVs and MAVs. Future work will include simulation of dynamic mobile nodes, allowing for more realistic real-life simulations. I also plan to extend the simulator to more complex scenarios, such as
those where the environment size is initially unknown or the search field is a vast area. Additionally, the algorithm is currently based on a centralized algorithm. The next research and development step will include distributed search and localization algorithm. There are also plans to implement this algorithm as a real-life experiment using the Qball MAVs [62] in the MAVSeN laboratory at Louisiana Tech University. In addition to an implementation, I also plan to improve the phase-machine, especially in PADF efficiency and emitter localization accuracy.
CHAPTER 5

MICRO-AERIAL VEHICLES AND SENSOR NETWORKS LABORATORY

5.1 Introduction

WSN have the ability to form ad-hoc networks and the nodes that these networks are composed of are capable of being interfaced with various sensors or controlling actuator systems, physiological status of troops or monitoring areas for insurgent activity, and the detection of radiological or biological agents. For this very reason, the Micro Aerial Vehicles and Sensor Networks (MAVSeN) lab has been designed specifically for the purpose of research in small-scale aerial vehicle design, cooperative intelligent sensing, and control algorithms of such platforms for various applications. The MAVSeN lab, though not the first of its kind—as other universities [63,64] and research facilities, such as MIT [65] and the Air Force Research Lab, respectively, have comparable infrastructures—has the capabilities to perform a myriad of experiments in various research fields, from swarm behavior to autonomous target tracking. Thus, the MAVSeN laboratory is designed to experiment with swarms of MAVs and sensor networks, in both layered and cooperative sensing concepts. The laboratory setup provides a high-speed and high-resolution motion capture system that
emulates indoor GPS. The motion capture system has the ability to track any objects, outfitted with specialized fluorescent markers, visible to the network of cameras.

5.2 Laboratory Functionality and Setup

The MAVSeN laboratory provides unique capabilities for experimenting with multiple MAVs and sensor networks, using layered and cooperative sensing concepts. The laboratory setup consists of a high-speed and high-resolution motion capture system that emulates indoor GPS, conceptually shown in Figure 5.1.

![Conceptual Micro-Aerial Vehicle and Sensor Network (MAVSeN) Laboratory.](attachment://figure5_1.png)

The motion capture system has the ability to track objects via high precision tracking data with slow or high speed motion using markers of sizes ranging from 3 to 25mm in diameter, seen in Figure 5.2. Tracking such small objects would be
conducive to research on significantly smaller MAVs akin to the interests of the Air
Force: bird-sized MAVs with flapping wings or smaller autonomous airborne vehicles.
The laboratory consists of a Vicon motion tracking system, various MAVs and wireless
sensor network platforms that can be used in experiments related to cooperative
control, detection, reconnaissance, and monitoring applications.

Figure 5.2: Fluorescent Tracking Marker—3mm, 9mm, and 14mm markers, from left
to right

5.2.1 Command and Control Network

The MAVSeN’s command and control network is composed of two separate
wireless data streams that send and receive data in real-time, as shown in Figure 5.3.
The first component is the position information from the MX T-40 cameras. This
information emulates GPS system. Giganet switch transfers camera data to the server.
The Vicon Tracker software processes the data, then transfers it through an internal
port connection to a multi-client server. Data frames containing object data are polled
from the Vicon Tracker. Each tracking object in the camera’s field of view is defined
as an object and is composed of four markers configured in a non-symmetric planar
topology. The server iterates through all four markers that define the object being
tracked and packetizes this information. Each MAV is configured as a client and upon model execution connects to the Vicon server.

![Diagram of data stream between motion capture cameras and MAVs.](image)

Figure 5.3: Data stream between motion capture cameras and MAVs.

The second component of the command and control network is the control station that controls the MAVs, in this case Quanser ground station. The ground station is responsible for uploading compiled code, transmitting joystick commands, and receiving flight system data. The Quanser ground station enables users to design and edit Simulink/QuaRC control system models. Embedded code that runs on the on-board Gumstix modules is created from Simulink as embedded C through the QuaRC software library, then uploaded wirelessly to the Gumstix. Model activation is performed by running the model in QuaRC and execution of the model occurs when
the user increases the throttle position on the USB joystick above a certain threshold. As a safety mechanism, values below that threshold cause the flyer to land.

In addition to fully autonomous flight execution, each MAV can be flown tele-operationally with models that translate the USB joystick control inputs and transmit them directly to the Gumstix from the ground station. Tele-operation models can be configured for full user control or partial user control. Partial user control allows one or more control surfaces to be actively actuated by the control system to maintain user-defined set points for height, roll, and pitch commands.

5.2.2 Vicon Motion Capture System

The camera system consists of ten Vicon T-40 cameras (one such camera is shown in Figure 5.4). Each camera, configured with 12.5mm lenses that give an effective Field of View (FoV) of 66.7 by 51.6 degrees at a 5m camera distance, is outfitted with a CCD sensor having a resolution of $2,352 \times 1,728$ (4,064,256) pixels [66]. This provides a full resolution maximum capture rate of 370 fps for each camera. Surrounding each lens is a circular array of 252 near infrared (780nm) LEDs, which are adjustable in brightness to increase the sensitivity of the cameras for different lighting conditions. This enables the system to provide high precision tracking data with slow or high-speed motion using markers of sizes ranging from 3mm to 25mm in diameter.

Markers are attached to objects that are to be tracked inside the capture volume. The versatility of the Vicon MX T-40 camera enables tracking of markers as small as 14mm in volumes $10m \times 10m \times 4m$. Cameras are networked via gigabit cables
that provide power, control, and data transfer from the Vicon MX Giganet switch. The Giganet switch has connectivity for up to ten cameras and can be networked with additional switches if more cameras are needed. The captured camera data is then fed into the Vicon server and displayed using the Vicon Tracker software for real-time analysis and capture. This data can also be transferred to the Quanser Workstation for flight algorithm position analysis in MATLAB, Simulink, and QuaRC via a crossover cable connection.

Figure 5.4: Vicon T-40 camera.
CHAPTER 6

POSITION-ADAPTIVE ALGORITHM FOR MAV CONTROL

6.1 Introduction

Localization using sensor networks and MAVs or UAVs has been researched primarily in missions of search and exploration [7,16,21]. While each mission has its own unique objectives, a common thread is navigation in the environment. Navigation can be accomplished autonomously or semi-autonomously through control algorithms [8], or completely by human interaction with some external feedback, i.e., sensors readings or a video stream. In either scenario, a combination of important factors such as localization techniques, path planning, and obstacle avoidance must be addressed. Considering the PADF technique, in which mobile sensors can adapt their position in order to improve localization accuracy, the adaptation refers to a sensor's location or position, not a modification of the control algorithm itself [67]. Most navigation algorithms rely on a vision-based navigation protocol or a pre-planned flight trajectory to traverse an environment. Vision-based methods can use images from a camera to map out the environment at detect obstacles. For instance, [68] presented a novel stereo-based obstacle avoidance system on a vision-guided MAV that was capable of fully autonomous maneuvers in unknown and dynamic environments. They used an obstacle
mapping algorithm to process stereo images, producing a 3-D map representation of
the environment using virtual scan, while a path planning algorithm rapidly computed
a suboptimal path. Other vision-aided research uses cameras to operate in real-time
as part of the navigation process without building maps [69,70].

Without the use of cameras one must rely on use of sensors for environmental
feedback—typically including the use of additional sensors, such as laser range finders
or sonar [71,72], or by more advanced techniques, such as simultaneous localization and
mapping, which uses the vehicle sensors to build a map of the environment [73]. Once
the environment is mapped out, algorithms such as the strap down inertial navigation
system [74], or the cooperative operations in urban terrain program [75] can be used
to investigate and develop the cooperative control algorithms needed for navigation
and obstacle avoidance. Path planning can be used instead of real-time modification
methods. In [72], a path planning module that combined two algorithms—an A* search
algorithm and a potential field method (PFM) was implemented. The A* algorithm
finds the shortest collision-free path from the current (estimated) position of the MAV
toward the target position and provided way-points that served as intermediate goals
for the PFM. The PFM then calculated a feasible path from the current position
to the farthest way point within a line of sight from the MAV. Unlike any of these
discussed methods, in the PADF method MAVs, using EM sensors, move based on
a cost function that improves the index of performance as it transitions from one
position to another.
6.2 Position-Adaptive MAV

Using the results from the static experiments for position adaptation algorithm development, we developed an autonomous control of a mobile node based on the cross PLE of the surrounding receivers combined with the RSSI from mobile node to the target that will allow the mobile node to move from places of lower estimation accuracy to places of higher estimation accuracy. For instance, in Figure 6.1, based on the positions of the receivers, the mobile node will try to move from the area of clustered nodes, to an area where it is positioned in front of the leakage point while maintaining line-of-sight with other receivers.

![Figure 6.1: Adaptive behavior projection.](image)

6.2.1 Algorithm Development

**Assumption 7.** There is a single stationary hidden target concealed by an obstruction, with a portion of the obstruction clear as a leakage point.

**Assumption 8.** All receivers are of known location and deployed in the same environment as the target.
Given those assumptions, let the \( 	ext{RSS}[(X_M, Y_M), (x_T, y_T)] \) be the received signal strength at location \((X_M, Y_M)\) of the mobile node from the signal transmitted by the target emitter \((x_T, y_T)\). We then define the path loss exponent between two receivers as \( \text{PLE}[(X_M, Y_M), (x_k, y_k)] \), where \((x_k, y_k)\) is a static receiver node’s position. Based on these, we propose an adaptive behavior where the mobile node moves such that the RSS between the mobile node and the target increases, while the mean cross PLE among all receivers (including mobile node) decreases. This can be better summarized as a cost function:

\[
J = Q \frac{1}{\text{RSS}[(X_M, Y_M), (x_T, y_T)]} + R \frac{1}{N} \sum_{k=1}^{N} \frac{\text{PLE}[(X_M, Y_M), (x_k, y_k)]}{N},
\]

where both \( Q \) and \( R \) are design parameters that adjust thresholds for RSSI and PLE, respectively. The adaptive policy should follow a path that minimizes \( J \), i.e., adjust \((X_M, Y_M)\) such that \( J \) is minimized via a simple discrete version of the gradient descent algorithm—the “hill climbing” optimization technique. Hill climbing is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found. Thus, the new positions \((X_{M_{\text{new}}}, Y_{M_{\text{new}}})\) can be calculated as

**Step 1:**

\[
J_{\text{old}} = Q \frac{1}{\text{RSS}[(X_M, Y_M), (x_T, y_T)]} + R \frac{1}{N} \sum_{k=1}^{N} \frac{\text{PLE}[(X_M, Y_M), (x_k, y_k)]}{N}.
\]

**Step 2:**

Calculate new \( J \) value, called \( J_N \) for all surrounding cells, i.e., cells that border
the current cell (cell with mobile node’s position of $(X_M, Y_M)$) to the north, south, east, and west by a single step unit.

$$J_N = Q \frac{1}{RSS \left[(X_M \pm \delta, Y_M \pm \Delta), (x_T, y_T)\right]} + R \sum_{k=1}^{N} \frac{PLE \left[(X_M \pm \delta, Y_M \pm \Delta), (x_k, y_k)\right]}{N}.$$ 

**Step 3:**

Choose minimum value of four cases for $J_N$, called $\text{minVal}$, to be compared to $J_{old}$.

**Step 4:**

*If* $(\text{minVal} < J_{old})$

$$X_{M_{\text{new}}} = x_{-\text{position}}[\text{minVal}],$$

$$Y_{M_{\text{new}}} = y_{-\text{position}}[\text{minVal}].$$

*Else*

$$X_{M_{\text{new}}} = X_M,$$

$$Y_{M_{\text{new}}} = Y_M.$$ 

Continue Step 4 for as long as $\text{minVal} < J_{old}$.

Note, that in scenarios where there are multiple mobile agents, the above cost function would result in the agents converging towards the same point, if all are based on the same position-adaptive algorithm. Therefore, to accommodate for additional mobile agents, we propose a cost function that has a repelling force between mobile sensor nodes, thus keeping them away from each other. Let $\mathbb{M}$ be a set of all mobile sensor node IDs. Then, the cost function that governs the motion of multiple mobile
sensor nodes is given by

\[ J = P \sum_{i,j \in M} \frac{1}{\text{dist}((X_i, Y_i), (X_j, Y_j))} + Q \sum_{i \in M} \frac{1}{\text{RSS}((X_i, Y_i), (x_T, y_T))} + R \sum_{i \in M} \sum_{k=1}^{N} \frac{\text{PLE}((X_i, Y_i), (x_k, y_k))}{N}. \]  \( (6.2) \)

The first term here is a repelling force between MAV sensor nodes. Note that using the first term alone would cause all mobile nodes to move away from each other towards infinity. Therefore, the second term is necessary to counter balance the repelling force caused by the first component of the cost function. It is therefore assumed that all mobile nodes can sense the RSS from the hidden emitter. The parameter \( P \) is a weighting factor for spatial distribution of mobile nodes, the parameter \( Q \) is a weighting factor related to the RSS from the hidden emitter, and the parameter \( R \) is the weighting factor related to the cross-PLE between mobile and static sensor nodes.

### 6.2.2 Path-Loss Exponent

In Equations 3.7 and 3.8 the RSSI was calculated based on the log-normal shadowing model, specifically for IRIS WSNs. Considering these equations, where \( \alpha \) is the PLE, distance calculations that estimate each nodes' distance from the emitter are contingent on PLE. The PLE depends on the medium of EM signal propagation and is usually not known a priori. PLE numbers typically range from 2 to 5, with the value 2 being associated with free-space propagation (usually reserved for propagation in anechoic chambers or very short distances), and 4 being a distorted environment (such as an urban scenario with multiple buildings). We attempted to calibrate the PLE values pertinent to our laboratory setup. This not only served the purpose of
PLE calibration, but it also was necessary for accurate fabrication of RSS values implemented in a simulation for the position adaptive algorithm simulation.

6.3 Experimental PLE Calibration

In order to use PLE as a metric for position adaptation, we first calibrate the PLE in a laboratory environment. This experiment was performed inside of Louisiana Tech Universitys MAVSeN laboratory, as described in Chapter 5. The calibration was done for two separate scenarios—a proposed free-space environment and another in which the emitter is obstructed, simulating a distorted environment.

6.3.1 PLE Calibration in Obstruction-Free Environment

To calibrate the PLE, we first measured the RSSI at a reference distance of one meter, defined as $RSSI_0$ (value approximately 45). This value was converted to dBm (in this case, dBm value of -46) and is used according to Equation 3.8. The log-normal model states that the received power is not uniform when measured at different locations while maintaining the same distance separation between emitter and receiver; therefore, we collected multiple RSSI values produced from similar distances from the emitter. The distances ranged from one to six meters. Because RSSI is very erratic and unstable at times, 50 samples were collected and averaged for each distance measurement. Distances were recorded multiple times from different orientations, again, to include the unpredictability of RSSI values. These values were recorded in Table 6.1.
Table 6.1: Sample RSSI values based on varying distance.

<table>
<thead>
<tr>
<th>Emitter Position</th>
<th>Receiver Position</th>
<th>Distance (m)</th>
<th>Log(D)</th>
<th>RSSI (avg)</th>
<th>RSSI to dBm</th>
<th>$P_0$ (dBm)</th>
<th>$P_{ij} - P_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-2</td>
<td>1</td>
<td>0.000</td>
<td>47</td>
<td>-44</td>
</tr>
<tr>
<td>0</td>
<td>-2</td>
<td>1</td>
<td>-2</td>
<td>1</td>
<td>0.000</td>
<td>46</td>
<td>-45</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>-2</td>
<td>1.41</td>
<td>0.149</td>
<td>38</td>
<td>-53</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>2</td>
<td>0.301</td>
<td>44</td>
<td>-47</td>
</tr>
<tr>
<td>-1</td>
<td>-2</td>
<td>1</td>
<td>-2</td>
<td>2</td>
<td>0.301</td>
<td>42</td>
<td>-49</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>2.24</td>
<td>0.350</td>
<td>35</td>
<td>-56</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-2</td>
<td>2.24</td>
<td>0.350</td>
<td>38</td>
<td>-53</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>2.83</td>
<td>0.452</td>
<td>36</td>
<td>-55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-2</td>
<td>3</td>
<td>0.477</td>
<td>40</td>
<td>-51</td>
</tr>
<tr>
<td>-2</td>
<td>-2</td>
<td>1</td>
<td>-2</td>
<td>3</td>
<td>0.477</td>
<td>37</td>
<td>-54</td>
</tr>
</tbody>
</table>

These values were then plotted as dBm value versus log(D), as shown in Figure 6.2. The final dBm value was calculated by subtracting the dBm value of receiver to emitter ($RSSI - 91$) from the reference distance dBm ($P_0 - 91$). One can see by the variance in values at similar distances how RSSI can fluctuate. This variance is consolidated into a linear regression model that shows the trend of RSS at certain distances. This is shown as the solid line in the Figure 6.2, and the slope is given by

$$m = \frac{(RSSI - 91) - (P_0 - 91)}{\log(D)}, \quad (6.3)$$

and can be quantified in the bold linear equation (slope = -20.995). Deriving Equation 6.3 from Equation 3.7 yields

$$\frac{(RSSI - 91) - (P_0 - 91)}{\log(D)} = -10 \cdot \alpha = m \quad (6.4)$$
The value of PLE is approximately 2.1, suggesting free-space path loss and would seem contradictory with propagation being inside a laboratory with many multi-path losses. However, this small PLE is due to short distances incorporated in calibration.

![Experimental RSSI calibration results for unobstructed emitter.](image)

**Figure 6.2:** Experimental RSSI calibration results for unobstructed emitter.

### 6.3.2 PLE Calibration with Obstruction

Performing the same test as in Section 6.3.1, the RSSI was measured and recorded at multiple distances between emitter and receiver. However, this time, the emitter was placed in a foil-lined box that restricted EM propagation, except through the leakage point—a large opening or window in the front of the box. This can be
seen in Figure 6.3. Multiple distances were again taken into consideration. While the majority of RSS-to-distance measurements were taken behind or on the side of the box, there were multiple instances of the measurement being taken in front of the box. This would increase the RSSI value, which made for a more realistic and accurate depiction of the scenario that an obstruction would create.

![Emitter inside obstruction during RSS testing.](image)

The reference measurement at one meter, $P_o$, was taken behind the box away from the leakage point. The RSSI value recorded was 38 and was again converted to dBm (-53). The various RSSI data points measured for distance were recorded in a table and plotted, as shown in Figure 6.4. Based on a linear regression model, the slope was measured and used to calculate the PLE, which gives a value of 2.49. These
PLE models were used in the development of the MAV position control algorithm simulator.

Figure 6.4: RSSI calibration testing with emitter obstructed experimentation results.

6.4 MATLAB Simulation Development and Results

To test the functionality of the algorithm, we developed a MATLAB simulation tasked with taking multiple static receivers in conjunction with a single mobile node and target.

6.4.1 Development Parameters

In development of the simulation, we took into account how we wanted to address the problem. Therefore, initial assumptions were taken into consideration. Also, other assumptions had to be made that corresponded to simulation environment.

Assumption 9. The obstruction is a bounded, convex two-dimensional object.
**Assumption 10.** The minimum distance between the nearest neighbors of two separate obstructions is greater than the width of the mobile agent such that any planned path between the objects is navigable.

**Assumption 11.** The mobile node will be in close proximity to other static nodes. Based on previous research and experimental results, satisfactory localization accuracy was achieved when receiver nodes are surrounding the emitter with a relatively equal spacing from the emitter [12,29,30,76]. Any cluster of receivers will result in reduced accuracy, no matter if they are in front of leakage point or not. Leakage point placement does help reduce error, however, as nodes closer to front of box tend to produce a slightly better result. Therefore, the objective of the mobile node is to move from a position of cluster, to an area that would produce a better localization accuracy based on the control algorithm implemented.

### 6.4.2 Simulation Environment

The simulator was created in MATLAB to emulate the behavior of the adaptive position algorithm described in Section 6.2.1. It has the ability to simulate multiple receivers or static nodes. Based on given assumptions and constraints, any obstacle can be built in the simulation environment, as long as it consists of straight-edges. As shown in Figure 6.5, a sample of the GUI is shown with multiple static nodes (orange in color), a single target (red), a single mobile node (cyan), and an obstacle (solid black line) that happens to represent a “wall” as a single line. The search environment itself is seen as a grid (dotted lines) that help to incorporate the step size of the mobile node. The smaller the grid size, then the smaller steps the mobile node takes as it...
traverses from cell to cell. Movement from cell to cell is based on the position adaptive algorithm. The node chooses the next \( x_{m_{\text{new}}}, y_{m_{\text{new}}} \) from surrounding cells based on discrete time function described in Section 6.2.1.

![MATLAB simulator diagram](image)

Figure 6.5: Screenshot of MATLAB simulator for adaptive position algorithm.

The PLE calibration testing, performed in Section 6.3.1, was the basis for PLE and RSSI values in the simulation. The linear regression equations (bold equations in Figure 6.2 and Figure 6.4) used to calculate PLE to represent free-space path loss and obstacle-obstructed path loss, values of 2.1 and 2.5, respectively, are used to reverse-calculate approximate RSSI based on distance of cell position to target position. For example, for a cell position that is 10 meters from the actual location of the target nodes cell position with no obstruction in between, the RSSI at that sample node would be:
\[ RSSI_{ij} = (RSSI_0 + K) - 10 \cdot a \cdot \log(D) - K \]

\[ = (45 - 91) - 10 \cdot 2.1 \cdot \log 10 + 91 \]

\[ = 24. \]

These values are calculated on a per-case basis in the simulator depending on if an obstacle exists between the cell and the actual position of the target. A random noise with a S/N of 8:1 was added to values of the RSS. This accounts for a lower signal-to-noise ratio when the receiver is further from the target. The PLE between all surrounding receivers and the mobile node, denoted as CrossPLE, was calculated as the mean of the individual PLE values. Each PLE value was derived from the PLE calibration experiment—either 2.1, suggesting free space, or 2.5, suggesting an object is interfering with propagation. The design parameters, \( Q \) and \( R \), are manually adjusted for RSSI and PLE sensitivity, respectively. As either value increases, it influences the position adaptive algorithm’s index of performance to go towards the target to increase RSS or move towards an unobstructed area to achieve line-of-sight with the adjacent static nodes.

6.4.3 Simulation Results

The position-adaptive algorithm was tested in a scenario in which five static nodes surrounded a static target. A mobile node is placed in the near vicinity of a static node. A single object is added in the search environment. The objective is for the mobile node to move to an area of higher localization accuracy based on the cost function 6.1. The parameter \( Q \) represents the weighting factor related to the node's
closeness to the target, and the parameter $R$ represents the weighting factor related to mobile node's direct line-of-sight with other static nodes (important in order to keep good communication).

**Scenario 1: Wall-Type Obstacle**

In the first experimental simulation, the environment was set up as described above with a "wall-like" obstacle between several of the static nodes and the target, as shown in Figure 6.6. The mobile node starts in the upper-left corner near the static node. I varied the values of the control parameters and plotted the trajectory that the mobile node took. The simulation is finished when the cost function, $J$, was minimized. Note that as the mobile node encountered an obstacle (placed between current positions and desired next position), the algorithm chooses the next best position, excluding previous position. The simulator was run five times for different values of parameters $R$ and $Q$. In the figure below, the paths for the different values are colored and each path termination is signified by the white square and $R$–$Q$ ratio. Notice the fifth path with the highest value of $Q$, representing RSSI as prime factor, came closest to the target; whereas the third path, with the highest $R$, stayed in line-of-sight of nodes 1, 2, and 5.
Figure 6.6: Simulation with single object as a “wall”. The simulation was run five times for various values of R and Q, thus changing the trajectory/path of the mobile node.

**Scenario 2: Box-Type Obstacle**

Similar to Scenario 1, the simulation was set up with analogous placement of static nodes, target, and initial starting position of mobile node. However, in this scenario, the target was encased in a “box-like” obstacle similar to Figure 6.3. Again, the objective was for the mobile node to move to positions that would achieve potential higher localization accuracy. In Figure 6.7, as Q continues to increase, the paths get closer to the box’s front, which is considered a leakage point and will produce the cleanest levels of RSSI, thus making accuracy higher than if taken behind the box. Note that in the path2 case, because the values of Q and R are low, there is
no incentive for a mobile node to move right such as in other cases and the node stays within direct line-of-sight with static nodes 1 and 2. While there is an infinite number of scenarios possible to create, these two generic simulations show that the position adaptive algorithm will indeed traverse the search environment based on the cost function. Adjusting the design parameters will change the path or trajectory of the mobile node contingent on the mission requirements.

Figure 6.7: Simulation with target encased in “box”. The object restricted EM propagation, except to front of box, thus creating a leakage point. The simulation was run five times for various values of $Q$ and $R$. 
6.5 Conclusion

I presented a cost function that is based on PLE and RSSI mobile and static nodes in a deployed environment. The method is based on a position-adaptive direction finding concept where the mobile node moves to positions of higher localization accuracy. I allow for multiple cooperative mobile nodes and more than a single target to be included. The algorithm, though constrained to initial assumptions, can be implemented in a real-time experiment with MAV-quadrotors at the Louisiana Tech University's MAVSeN lab. In the future I will investigate the use of complex cepstrum between mobile nodes and the hidden emitter as a metric for a MAV control and obstacle avoidance. Complex cepstrum is correlated with a received echo in EM signals. Therefore, reducing the cepstrum leads to improving measurement and estimation in an environment (similar to phenomenon in biological species with built-in cepstrum minimization and detection, or echo reduction, capabilities that allow movement away from large objects/animals). Preliminary results have shown that while doing the complex cepstrum of a signal, any peaks or significant spikes observed in the analysis can be interpreted as an echo that corresponds to an obstacle in the EM propagation environment. Knowing the proximal locations of obstructions allows the metric to adjust the position of the mobile nodes in order to improve localization accuracy.
CHAPTER 7

PADF IMPLEMENTATION WITH MAVS AND WSNS

Using a combination of hardware (IRIS wireless sensor nodes) and software modules (MATLAB and Java), both a static and mobile localization experiment was performed to test the accuracy of finding an unknown emitter. The IRIS nodes are capable of both the communication and sensing capacity. A brief description of the architecture and design of these nodes is given below.

7.1 Hardware Components

IRIS Wireless Sensor Nodes: The IRIS 2.4 GHz mote by MEMSIC, shown in Figure 7.1, is a module used for enabling low-power, wireless sensor networks [56] by use of the embedded system, TinyOS, written in the NesC language [77]. It has a 2.4 to 2.48 GHz globally compatible ISM band; a 250 kbps high data rate radio (outdoor line-of-sight tests yielded ranges as far as 500 meters between nodes without amplification); an IEEE 802.15.4 compliant RF transceiver; and a direct sequence spread spectrum radio (resistant to RF interference PROVIDES inherent data security.) A ground antenna plane was added that shields the antenna from the rest of the node electronics, preventing interference with RSSI measurements. This creates more
omni-directional measurements, which is very important for integration with ground vehicles and MAVs, as seen on the rightmost side of Figure 7.1.

The primary function of the receiver sensor nodes is to detect the EM source and transmit the RSSI values between neighboring nodes and the emitter. The RSSI values are being collected at a rate of ten samples per second, or every 100 milliseconds, and then forwarded to the base station, where a host computer collects all data and calculates the position of the emitter using a localization algorithm based on LSE.

We tested the concept of estimating an emitter's position using RSSI values with IRIS sensor nodes. To add to the complexity of the problem and make it more conducive to a real-world scenario, such as one pertaining to an urban environment, the emitter was placed in a foil-lined box to restrict EM propagation, except from the opening in the front (which was considered a leakage point). The following sections describe the testing using IRIS wireless sensor nodes.

**Qball MAVs:** The MAVSeN laboratory has three Qball-X4 platforms, shown in Figure 7.2. The Qball-X4 is a rotary wing vehicle platform developed by Quanser
that provides a suitable MAV platform for a variety of research applications. It is a quadrotor design with four motors fitted with 10-inch propellers. The entire mechanism is enclosed within a protective carbon fiber cage, making it an ideal tool for basic vehicle navigation and control. The MAV is equipped with QuaRC software real-time control and multi-agent mission development frameworks; a ground control station; and embedded computer systems and inertial measurement units.

Porting the cooperative control algorithms can be done through the software development toolkit. We integrate Qball-X4 MAVs with IRIS nodes for mobile sensing applications. The MAVSeN laboratory also has one DraganFlyer X-6 GPS guided UAV helicopter capable of outdoor flights. Qballs are controlled by full-featured embedded avionics data acquisition card, the Quanser HiQ, which provides high-resolution inertial measurement sensors and motor outputs. High-level control of the Qball is performed by the Gumstix Verdex embedded computer platform, which has been configured as a QuaRC target system. Wireless communication to the ground station and other vehicles can be configured to be either IEEE 802.11 (WiFi) or IEEE 802.15.4 (ZigBee) [62].
7.2 Localization Experiments with Unobstructed Emitter

The objective of the localization experiments performed in the laboratory environment was to achieve the following goals:

- test accuracy of the localization algorithm,
- examine stability and sensitivity of given configurations,
- explore propagation of the EM signal in embedded environment with leakage points,
- define a metric that corresponds to a given configuration’s localization estimate.

As such, we sought to obtain accurate emitter localization results using various random and controlled configurations that would incorporate different RSSI values at different positions. For this particular experiment, four IRIS nodes were used as receivers, with one additional IRIS node used as a hidden emitter. One particular configuration was called the “diamond” configuration—for its shape being similar to
that of a geometric diamond in terms of receiver node positioning. An example of the
diamond configuration is shown below in Figure 7.3. Notice that all of the receiving
nodes surround the transmitter equilaterally. Each receiver’s distance estimate, based
on RSSI measurements, can be modeled as a “disc” around the node. The LSE
uses the intersections of these disc areas to determine the position of the transmitter.
Figure 7.4 shows a MATLAB visualization of the area of the intersecting discs and
the actual location of the transmitter.

Figure 7.3: Receiving sensor nodes in a diamond configuration surrounding
transmitting node.
Figure 7.4: An intersection of circles with radii equal to RSS measured at individual nodes; width of disks simulate RSS uncertainty.

There are four other user-defined configurations, shown in Figure 7.5 (topologies 2-5, clockwise), as well as multiple random positions that were studied in this experiment.

Figure 7.5: Configurations or topologies 2-5, from top left, clockwise.
7.2.1 Experimental Results for Static WSN with an Unobstructed Emitter

The accuracy of the estimation was determined by comparing the distance between the estimated and actual positions of the emitter. The positions are given as coordinates as based on a Cartesian system. The emitter was placed at the origin, (0, 0). After testing the initial five different configurations, the “diamond” configuration had one of the highest accuracy rates, as shown in Table 7.1. The first three rows, which refers to the “diamond” configuration and those closely related to it shown in Figures 7.3 and 7.5, respectively, have the lowest localization error in terms of estimated position versus actual position of the hidden emitter. This is an expected result since LSE is based on the intersection of the RSSI radii from all receiving nodes, e.g., Figure 7.4. All distances are given in meters.

Table 7.1: IRIS platform for multiple configurations with error (case 1-5 corresponds to configurations 1-5 from Figure 7.3 and Figure 7.5)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>X-pos</th>
<th>Y-pos</th>
<th>X-pos</th>
<th>Y-pos</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0563</td>
<td>0.0117</td>
<td>0.0174</td>
<td>0.0195</td>
<td>0.0396</td>
</tr>
<tr>
<td>2</td>
<td>-0.1341</td>
<td>-0.0271</td>
<td>0.0174</td>
<td>0.0195</td>
<td>0.1368</td>
</tr>
<tr>
<td>3</td>
<td>0.0063</td>
<td>-0.0805</td>
<td>0.0174</td>
<td>0.0195</td>
<td>0.101</td>
</tr>
<tr>
<td>4</td>
<td>-0.0094</td>
<td>-2.1713</td>
<td>0.0174</td>
<td>0.0195</td>
<td>2.17</td>
</tr>
<tr>
<td>5</td>
<td>0.0373</td>
<td>-2.5462</td>
<td>0.0174</td>
<td>0.0195</td>
<td>2.56</td>
</tr>
</tbody>
</table>

When the receivers are relatively equally distributed around the emitter, to varying degrees of distance and configuration, we get higher estimation accuracy, which has also been shown in previous research [12,76]. Any cluster of all receivers will greatly degrade the accuracy of the localization. This clustering was tested in more detail in the next section involving obstructions.
Any cluster of all receivers will greatly degrade the accuracy of the localization. This clustering was tested in more detail in the next section involving obstructions. Even when spatially distributed, it is important to note that distance plays a factor on the degree of accuracy as well. The same configuration can produce varying accuracy depending on how close the receivers revolve around the emitter. Table 7.2 shows the error when testing distance between the receiver and the emitter for "diamond" configuration. This particular configuration was emphasized because of its initial accuracy in the localization experiment. Based on the experimentation, if the receiver is too far from the emitter, then the RSSI values are too low or vary too much to produce accurate results [27]; however, if too close, then near-field antennae interruptions can occur, causing distorted RSSI values.

Table 7.2: IRIS platform, diamond configuration, with varying distance.

<table>
<thead>
<tr>
<th>Distance (m)</th>
<th>Estimation</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X-pos</td>
<td>Y-pos</td>
</tr>
<tr>
<td>1</td>
<td>0.189</td>
<td>2.208</td>
</tr>
<tr>
<td>1.5</td>
<td>-0.260</td>
<td>2.866</td>
</tr>
<tr>
<td>2</td>
<td>0.064</td>
<td>2.200</td>
</tr>
<tr>
<td>2.5</td>
<td>0.025</td>
<td>2.303</td>
</tr>
<tr>
<td>3</td>
<td>-0.050</td>
<td>2.311</td>
</tr>
<tr>
<td>3.5</td>
<td>0.044</td>
<td>2.317</td>
</tr>
<tr>
<td>5</td>
<td>0.043</td>
<td>2.572</td>
</tr>
<tr>
<td>8</td>
<td>0.010</td>
<td>2.483</td>
</tr>
<tr>
<td>10</td>
<td>-0.018</td>
<td>2.480</td>
</tr>
<tr>
<td>12</td>
<td>0.009</td>
<td>2.469</td>
</tr>
<tr>
<td>16</td>
<td>0.359</td>
<td>2.121</td>
</tr>
</tbody>
</table>
7.3 Localization Experiments with Shielded Emitter

Performing the same experiments as previously described for multiple receiving nodes surrounding a hidden emitter without obstruction, the emitter was placed in a foil-shielded container used to restrict EM propagation. This container was an 18 × 24 × 18-inch box covered on all sides with aluminum foil to shield or restrict the radio signal propagation. Approximately, a 144-square inch opening was created in this box as a leakage point. This creates a more challenging environment to detect the hidden emitter—similar to situations in an urban environment where there are may be various EM leakage points or leakage areas. The receiving sensor nodes were again placed around the emitter in five distinct configurations, upon which the emitter position was estimated.

We measured the response of the IRIS nodes and corresponding RSSI values. Each receiver is placed on a plastic stand about four feet high. It is expected that the degradation of the EM signal passing through the box would reduce the accuracy of the LSE position of the transmitter. This experimental setup is shown in Figure 7.6.
7.3.1 Results of Localization Experiment with Shielded Emitter

The same experiments were performed; however, this time the sensor nodes placed around the emitter housed in the foil-shielded container in a coordinated fashion. Upon testing, the numbers were skewed compared to the original results of Table 7.2, distinctly showing higher error than previous testing. This was attributed to the foil on all sides shielding the EM signal propagation and attenuating the RSSI values, except through the leakage point (window) in the front. By obtaining lower values, the receiving nodes sense they are further away from the target than they actually are, giving an inaccurate estimation subsequently. To compensate for this loss, we used the cross PLE of surrounding receivers and weighted the distance estimation, which enhanced the accuracy, as explained in Chapter 3.4.1.
These new results can be seen in Table 7.3. Similar to before, clustering of nodes, even in front of the box where leakage point gives better RSSI values, still degrades the localization accuracy.

Table 7.3: Multiple configurations with emitter hidden in a shielded container using weighted-position estimation algorithm.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Estimation</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X-pos</td>
<td>Y-pos</td>
</tr>
<tr>
<td>1</td>
<td>0.0432</td>
<td>-0.0743</td>
</tr>
<tr>
<td>2</td>
<td>-0.2366</td>
<td>-0.1041</td>
</tr>
<tr>
<td>3</td>
<td>-0.0592</td>
<td>-0.1427</td>
</tr>
<tr>
<td>4</td>
<td>-0.0094</td>
<td>-2.1713</td>
</tr>
<tr>
<td>5</td>
<td>-0.056</td>
<td>-2.4416</td>
</tr>
</tbody>
</table>

Upon testing, the values in Table 7.3 when compared to the original results of Table 7.2, distinctly showed higher error. This was attributed to the foil on all sides shielding the EM signal propagation and attenuating the RSSI values, except through the leakage point (window) in the front. By obtaining lower values, the receiving nodes sense they are further away from the target than they actually are, giving an inaccurate estimation subsequently. To compensate for this loss, I incorporated the cross-PLE of surrounding receivers and weighted the distance estimation algorithm, which enhanced the accuracy, as explained in Section 3.4.1. The difference between the weighted estimation algorithm and non-weighted algorithm can be seen in Table 7.4. One can see that for each case, save for configuration 4, the performance is at least twice the error rate than its predecessor when the weighting algorithm is not applied.
Table 7.4: Comparison of weighted algorithm versus non-weighted algorithm.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Error Weighted</th>
<th>Error Standard</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.107</td>
<td>1.0015</td>
<td>-10x</td>
</tr>
<tr>
<td>2</td>
<td>0.275</td>
<td>0.8817</td>
<td>-3x</td>
</tr>
<tr>
<td>3</td>
<td>0.157</td>
<td>1.8276</td>
<td>-11x</td>
</tr>
<tr>
<td>4</td>
<td>2.07</td>
<td>1.8903</td>
<td>+1.1x</td>
</tr>
<tr>
<td>5</td>
<td>2.435</td>
<td>4.1176</td>
<td>-2x</td>
</tr>
</tbody>
</table>

Similar to the previous configurations test, the clustering of nodes, even in front of the box where leakage point gives better RSSI values, still degrades the localization accuracy. This clustering effect was tested in a separate experiment in which multiple configurations were chosen at random, and then the localization algorithm was applied. Distance was constrained to four meters, and positions were chosen to test both spatial distribution and clustering. The varying positions were charted in Figure 7.7 and are separated by color and shape. For clarity sake, a cluster was defined as all four receivers placed in near vicinity of each other on any one particular side of the emitter. As evidenced in Table 7.5, configurations 3, 5, and 8, which correspond to the red triangle, cyan star, and magenta 'x' in Figure 7.7, respectively, show how clusters yield poor results, regardless of leakage point placement. Leakage point placement does help reduce error, however, as both configuration 6 and 7 (which are predominately near opening) both yield slightly better results opposed to configurations such as 2 and 4, which seem to be placed away from the opening.
Figure 7.7: Testing accuracy of position placement using random configurations of receivers.

Table 7.5: Error of position placement using random configurations of static receivers.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Cluster</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>0.2537</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>0.2631</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>2.2046</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>0.2344</td>
</tr>
<tr>
<td>5</td>
<td>Yes</td>
<td>1.8654</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>0.1501</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>0.1593</td>
</tr>
<tr>
<td>8</td>
<td>Yes</td>
<td>2.3633</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>0.2663</td>
</tr>
</tbody>
</table>
7.4 Experiments with MAV-based WSNs

In order to further investigate PADF concepts in an indoor environment, an experiment, as depicted in Figure 7.8, is developed for purposes of investigating potential refinements in consistency, sensitivity, and robustness via the design and implementation of three stationary platforms and one continuously moving platform during the localization phase.

![Figure 7.8: Experimental position adaptive direction finding (PADF) concept using MAVs with wireless sensor nodes.](image)

7.4.1 Sensitivity and Consistency Testing and Results

As shown in Figure 7.8, a MAV that is integrated with a custom sensor node for networked cooperative PADF is flown in a “zig-zag” trajectory around an experimental region-of-interest. The box is partially wrapped with aluminum foil and covered with blue cloth to avoid inducing ambiguities in the distributed camera-based optical navigation system. An emitter is embedded in the vicinity of a RF leakage point that is cut out from the front of the enclosure.
A complete data set is comprised of collecting series of measurements with all four receiving platforms while operating the MAV along an 8-point localized zig-zag pattern. The other three nodes are stationary. Figure 7.9 is the plot of the PADF metric (blue line), localization error (red line), and a fluctuating RSSI-based distance estimate (green line) from the fourth dynamic MAV to the embedded emitter. The metric itself was derived from the variance of cross-PLE values between the surrounding receivers. Uniform environments without obstacles will lead to similar values of cross-PLE, thus reducing the variance and the metric. Theoretically, the metric should mimic the error and provide a basis for receiver convolution in the environment. One can notice regions of relative stability associated with the distance.

![Figure 7.9: MAV lab tests: localization error and PADF metric for different experiments.](image-url)
In addition to exploring the sensitivity of PADF configurations, I also studied the effects of multi-path echoes and RSSI deflections on the localization accuracy. Experimental results show anomalies in the data that indicate very slight deflections due to the EM signal escaping the box. These deflections are very slight, and do little to change the RSSI peak value. As shown in Figure 7.10, the left portion, subfigure 7.10(a) shows the trajectory of a single mobile node (blue stars) moving around the emitter in the box (red star) in a semi-circular pattern. Notice in subfigure 7.10(b), the highest values of RSSI are near the middle of the chart because of the front opening (EM leakage point) in the box.

![Experiment #1](image)

(a) MAV trajectory, in blue  
(b) RSSI values (higher value in red)

Figure 7.10: RSSI peak value measurement.

To test the validity of the leakage point and RSSI measurements, another experiment was performed to exclude the leakage point. The single mobile node was moved around the back of the box, where no leakage point was evident. Figure 7.11 shows the MAV trajectory around the back of the box, where no window or leakage point exists. Because of the lack of a leakage point, the values are relatively equal.
with slight variances in value, as shown in subfigure 7.11(b). The values, however, indicate there is no significant spike in the RSSI values, which is to be expected.

![Graph of MAV trajectory and RSSI values](image)

Figure 7.11: RSSI peak value measurement from behind the box

### 7.4.2 Cepstrum Testing and Results

I also provide a complex cepstrum analysis that is commonly used for detecting a presence of echoes in received EM signals. This in turn can be used as a measure of quality of MAV positions—accurate estimation requires low echo in received wireless signals. Cepstrum of a signal \( s(n) \) is defined as the power spectrum of the logarithmic power spectrum and is given by

\[
c_s(n) = |\mathcal{F}\{\log(S(n))\}|^2. \tag{7.1}
\]

The complex cepstrum is the logarithmical-weighted components of the signal spectrum and is given by

\[
cc_s(n) = \text{IFFT} \{\log(S(n))\}. \tag{7.2}
\]
The independent variable of a cepstral graph is called the "quefrency", which is a measure of time, though not in the sense of a signal in the time domain. Since the complex cepstrum can be interpreted as representing the logarithmical-weighted components of the signal spectrum (i.e. right half of Figure 7.12), the sharp peaks in the middle and upper "quefrency" regions of the complex cepstrum (as a function of spatial trajectory coordinates) represent sensitivities and consistencies in complex multi-modal scattering environments over small localized portions of a platform trajectory.

![Figure 7.12: Analytical model for investigating the development of intelligent multi-path echo trend computations (cepstrum).](image)

By doing the complex cepstrum of a signal, any peaks in the analysis can be interpreted as an echo, which usually corresponds to an obstacle in EM propagation. I performed multiple tests on complex cepstrum analysis in a clear search environment versus an environment with a large obstacle. The visualization of the lab experiment and the setup can be seen in Figure 7.13. The obstacle used in this scenario was a large box lined with aluminum foil. This would restrict EM propagation the same way as the box containing the emitter. A zig-zag pattern was performed as shown.
on the left side of Figure 7.14 with eight distinct points. In the experiment with the obstruction, the mobile node had to navigate around the box (yellow box in image on right in Figure 7.14). I took the resulting RSSI measurements of 150 samples per distinct point and performed the complex cepstrum algorithm in Equation 7.2 on the data.
The complex cepstrum in Figure 7.15 shows that for the first experiment with no obstruction, the complex cepstrum at each distinct point shows no peaks. However, in the second experiment, with the large obstacle (box), points 1, 2, and 3 are hidden behind the box. Looking at the second set of graphs, Figure 7.16, points 1-3 are clearly disproportionate to points 4-8. This variance of data in the first three positions proves that the obstacle was indeed detected using the cepstrum analysis. As the object blocked the signal, the reverberations caused echoes that were shown on the graph.
Figure 7.15: Cepstrum results of experiment without obstruction.

Figure 7.16: Cepstrum result of experiment with obstruction.
CHAPTER 8

CONCLUSIONS

I have experimentally demonstrated that RSS can effectively be used for localization of a hidden emitter using wireless sensor networks. Localization accuracy is improved as the number of sensor nodes and their spatial distribution/spread increases. I have introduced the concept of position-adaptive direction finding applied to localization of a hidden, cooperative emitter where static and mobile nodes cooperate in the localization mission by coordinating their sensing, and adapting their position in real-time. I improved localization accuracy by adapting receivers’ positions to areas of higher estimated accuracy. I developed a custom three-phase machine for cooperative MAV control in large environments, that allows MAVs to search for target before ultimately localizing it. Additionally, I developed a simulator that implements a position-adaptive MAV control algorithm, based on RSSI and PLE between mobile and static nodes, in a deployed environment that will allow mobile nodes to reposition themselves for increased accuracy. This simulation can be used to simulate more complex scenarios as well as allow the design and implementation of other localization algorithms. In addition to simulation, I developed a laboratory capable of implementing PADF concepts in a real-world fashion through the use of MAVs and WSNs.
8.1 Future Work

The work presented in this dissertation tackled a very specific problem: using a single mobile agent along with multiple static agents to localize a hidden, stationary target. This research has real-world applications, such as habitat monitoring, in which an event can threaten the nature of the habitat. Still, a host of applications are viable by expanding this research. For instance, having explored the “position-adaptation” portion of PADF, I would like to explore the “direction-finding” segment by using directional antennas instead of omnidirectional antennas. This will allow MAVs to more efficiently search for higher values of RSS or locations of leakage points in urban environments. Similarly, other future work would include using multiple MAVs operating in a cooperative fashion, each performing the PADF technique and adapting their position for localization accuracy simultaneously. This process would be viable in scenarios such as search and rescue missions, where knowing the position of targets and being able to navigate and traverse environments could be vital.

In addition to multiple MAV agents, I intend to implement multiple targets, both static and mobile. Introducing mobile targets will require the implementation of cooperative control, collision avoidance, and formation control algorithms. I would also like to continue to improve the efficiency of the phase machine and implement the algorithm on multiple MAVs to be explored in a large domain. As a laboratory setup is too confined in space, a large area, such as a stadium or parking lot, can be used to spread MAVs from target, thus giving a challenge for search. GPS radios will have to be applied to the MAVs to get accurate position location. I plan to explore the use of a complex cepstrum analysis as feedback mechanism that can be used in
MAV repositioning and obstacle avoidance. Because the complex cepstrum senses large obstacles through echo detection, this can be used as a basis for the obstacle avoidance algorithms.

Versus working with a cooperative emitter, another challenge worth investigating is using a non-cooperative emitter with an unknown frequency in the localization problem. Using a custom, multi-frequency antenna is being developed to aid in this detection of unknown frequencies. This will prove useful in many commercial and military scenarios where RF signatures are unknown, such as discovery and localization of harmful devices i.e., IEDs and bombs, or tracking wireless signals in/around an area.
function varargout = PLEtest(varargin)
% PLETEST M-file for PLEtest.fig
% PLETEST, by itself, creates a new PLETEST or raises the
% existing singleton*.
% H = PLETEST returns the handle to a new PLETEST or the handle
% to the existing singleton*.
% PLETEST('CALLBACK', hObject, eventData, handles, ...) calls the
% local function named CALLBACK in PLETEST.M with the given input
% arguments.
% PLETEST('Property', 'Value', ...) creates a new PLETEST or raises
% the existing singleton*. Starting from the left, property
% value pairs are
% applied to the GUI before PLEtest_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property
% application
% stop. All inputs are passed to PLEtest_OpeningFcn via
% varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
% only one instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIDATA

% Edit the above text to modify the response to help PLEtest

%Author: Miguel Gates

% PLEtest.m is a simulator used to calculate PLE between receivers and
% target and use those values to reposition mobile node for better
% localization accuracy

% Last Modified by GUIDE v2.5 19-Mar-2013 12:15:46

% Begin initialization code - DO NOT EDIT
 gui_Singleton = 1;
 gui_State = struct( 'gui_Name',      mfilename, ...
                     'gui_Singleton', gui_Singleton, ...
                     'gui_OpeningFcn', @PLEtest_OpeningFcn, ...
                     'gui_OutputFcn',  @PLEtest_OutputFcn, ...
                     'gui_LayoutFcn', @PLEtest_LayoutFcn, ...
                     'gui_Callback', [ ] );

 if nargin & & ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
%
% End initialization code - DO NOT EDIT

% --- Executes just before PLEtest is made visible.
function PLEtest_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to PLEtest (see VARARGIN)

% Choose default command line output for PLEtest
handles.output = hObject;

% values to be used throughout
global start count J trail save_count;
J = 0;
save_count=0;
start = 1;
count = 1;
trail = [];
handles.intersection = [];
handles.x = [];
handles.y = [];
handles.found_obs = [];
handles.found_rec = [];
handles.data = [];
handles.mob_intersect = 0;
handles.previous = [0 0];
handles.distance = [];
handles.next = [];
%
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes PLEtest wait for user response (see.UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = PLEtest_OutputFcn(hObject, eventdata, handles)
    varargout cell array for returning output args (see VARARGOUT);
    hObject handle to figure
    eventdata reserved - to be defined in a future version of MATLAB
    handles structure with handles and user data (see GUIDATA)

    % Get default command line output from handles structure
    varargout{1} = handles.output;

% --- Executes on button press in receiver_push.
function receiver_push_Callback(hObject, eventdata, handles)
    hObject handle to receiver_push (see GCBO)
    eventdata reserved - to be defined in a future version of MATLAB
    handles structure with handles and user data (see GUIDATA)
    global node count;
    contents = get(handles.receiver_edit,'String');
    value = str2double(contents);
    start = count;
    finish = start + value - 1;

    axis on
    for i=start:finish;
        [node(i,1),node(i,2)] = ginput(1);
        labels = cellstr(num2str(i')) ;  % # labels correspond to order
        plot(node(i,1),node(i,2),'Color',[1 0.5 0.2],...  
        'MarkerSize',35,'Marker','.');    
        text(node(i,1),node(i,2),labels, 'VerticalAlignment','bottom',...  
        'HorizontalAlignment','right')
        count = finish + 1;
        axis([0 100 0 100]);
        hold on  
    end %legend([p1], 'Receiver', 'Location', 'Best');

function receiver_edit_Callback(hObject, eventdata, handles)
    hObject handle to receiver_edit (see GCBO)
    eventdata reserved - to be defined in a future version of MATLAB
    handles structure with handles and user data (see GUIDATA)

    % Hints: get(hObject,'String') returns contents of receiver_edit
    as text
str2double(get(hObject,'String')) returns contents of receiver_edit as a double

% --- Executes during object creation, after setting all properties.
function receiver_edit_CreateFcn(hObject, eventdata, handles)
    hObject handle to receiver_edit (see GCBO)
    eventdata reserved - to be defined in a future version of MATLAB
    handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
    See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

% --- Executes on button press in target_push.
function target_push_Callback(hObject, eventdata, handles)
    hObject handle to target_push (see GCBO)
    eventdata reserved - to be defined in a future version of MATLAB
    handles structure with handles and user data (see GUIDATA)
    global t_x t_y;
    test = findobj(gca,'Type','line','-and',' Color','r');
    test2 = findobj(gca,'Type','text','-and',' Color',[0 0 .1]);
    [t_x,t_y] = ginput(1);
    axis([0 100 0 100]);
    hold on
    plot(t_x,t_y,'r.',' MarkerSize',35);
    text(t_x,t_y, 'T', 'VerticalAlignment','bottom',...
        'HorizontalAlignment', 'right',' Color', [0 0 .1]);
    delete(test);
    delete(test2);

% --- Executes on button press in obstruct_push.
function obstruct_push_Callback(hObject, eventdata, handles)
    hObject handle to obstruct_push (see GCBO)
    eventdata reserved - to be defined in a future version of MATLAB
    handles structure with handles and user data (see GUIDATA)
global ob_x ob_y start;
temp = get(handles.ob_edit,'String');
obs = str2double(temp);
new_start = start;
finish = new_start + obs - 1;

axis([0 100 0 100]);
hold on;
for i = start:finish
    [t1,t2] = ginput(2);
    line([t1(1) t1(2)],[t2(1) t2(2)],'Marker','.','LineStyle','-','Color','k')
    ob_x(2*i-1:2*i) = t1;
    ob_y(2*i-1:2*i) = t2;
end
start = finish + 1;

% --- Executes on button press in mobile_push.
function mobile_push_Callback(hObject, eventdata, handles)
% hObject    handle to mobile_push (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global mob_x mob_y;
test = findobj(gca,'Type','line','-and','Color','c');
test2 = findobj(gca,'Type','text','-and','Color',[0 0 .2]);
[mob_x,mob_y] = ginput(1);
axis([0 100 0 100]);
hold on
plot(mob_x,mob_y,'c.','MarkerSize',35);
text(mob_x,mob_y, 'M', 'VerticalAlignment','bottom','HorizontalAlignment','right','Color', [0 0 .2]);
delete(test);
delete(test2);

% --- Executes on button press in grid_check.
function grid_check_Callback(hObject, eventdata, handles)
% hObject    handle to grid_check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of grid_check
on = get(handles.grid_check,'Value');
handles.sizeofCell = str2double(get(handles.cellsize_edit,'String'));

step = handles.sizeofCell;

pos(:,1) = [(step/2):step:100];
pos(:,2) = [(step/2):step:100];

axis([0 100 0 100]); hold on

if on == 1
    for i = 1:floor(100/step)
        line([pos(i,1)+(step/2) pos(i,1)+(step/2)], [0 100], ...
            'LineStyle', ':', 'Color', [.1 .1 .2], 'LineWidth', .8);
        line([0 100], [pos(i,1)+(step/2) pos(i,1)+(step/2)], ...
            'LineStyle', ':', 'Color', [.1 .1 .2], 'LineWidth', .8);
    end
else
    grid = findobj(gca, 'Type', 'line', '-and', 'Color', [.1 .1 .2]);
    delete(grid);
end

guidata(hObject, handles);

% --- Executes on button press in PLE_check.
function PLE_check_Callback(hObject, eventdata, handles)
% hObject    handle to PLE_check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of PLE_check
% exec

global pos t_x t_y mob_x mob_y;

on = get(handles.PLE_check,'Value');
%sizeofCell = str2double(get(handles.cellsize_edit,'String'));

if on == 1
    p(:,1) = [(sizeofCell/2):sizeofCell:100];
    p(:,2) = [(sizeofCell/2):sizeofCell:100];
    [t1,t2] = size(pos);
    obstacles = length(handles.found_obs);
    for i = 1:t1
        for j = 1:t1
            for k = 1:obstacles
yes_no(k) = check_intersect(pos(i,1), t_x, ...  
    handles.x(2*k-1), handles.x(2*k), pos(j,2), t_y, ...  
    handles.y(2*k-1), handles.y(2*k));
end
reality = sum(yes_no);
if reality >= 1;
    handles.data.pie(i,j) = 2.5;
else
    handles.data.pie(i,j) = 2.1;
end
end
end

%# labels correspond to their order
for i = l:tl
    for j = l:tl
        labels(i,j) = cellstr( num2str(handles.data.pie(i,j)));
        text(pos(i,1), pos(j,2), labels(i,j), ...
            'VerticalAlignment','bottom',...  
            'HorizontalAlignment','right', 'Color',[1 .4 .2],...
            'FontSize',6);
    end
end  
else
    test = findobj('Color',[1 .4 .2]);
    delete(test);
end
guidata(hObject, handles);

function r1_disp_Callback(hObject, eventdata, handles)
% hObject    handle to r1_disp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of r1_disp as text
% str2double(get(hObject,'String')) returns contents of r1_disp
% as a double
set(handles.r1_disp,'String','yes')
% --- Executes during object creation, after setting all properties.
function rl_disp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rl_disp (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
% called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in intersect_push.
function intersect_push_Callback(hObject, eventdata, handles)
% hObject    handle to intersect_push (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global node t_x t_y mob_x mob_y;

% Multi Receiver intersection test
handles.found_obs = findobj(gca,'Type','line','-and','Color','k');
inouter = length(handles.found_obs);
outer = length(node(:,1));
for i = 1:outer
    for j = 1:inner
        handles.x(2*j-1:2*j) = get(handles.found_obs(j),'Xdata');
        handles.y(2*j-1:2*j) = get(handles.found_obs(j),'Ydata');
        yes_no(j) = check_intersect(node(i,1), t_x, ...
            handles.x(2*j-1), handles.x(2*j), node(i,2), t_y, ...
            handles.y(2*j-1), handles.y(2*j));
    end
    reality(i) = sum(yes_no);

    if reality(i) >= 1 && i == 1
        handles.intersection(1) = 1;
    elseif reality(i) >= 1 && i == 2
        handles.intersection(2) = 1;
    elseif reality(i) >= 1 && i == 3
handles.intersection(3) = 1;

else
    handles.intersection(i) = 0;
end

end

% mobile node intersection test
for t = 1:inner
    yes_no2(t) = check_intersect(mob_x, t_x, handles.x(2*t-1), ...
                             handles.x(2*t), mob_y, t_y, handles.y(2*t-1), handles.y(2*t));
end

reality2 = sum(yes_no2);

if reality2 >= 1;
    handles.mob_intersect = 1;
else
    handles.mob_intersect = 0;
end

if outer == 1
    set(handles.r1_disp,'String', handles.intersection(1));
    set(handles.mobile_disp,'String', handles.mob_intersect);
elseif outer == 2
    set(handles.r2_disp,'String', handles.intersection(2));
    set(handles.r1_disp,'String', handles.intersection(1));
    set(handles.mobile Disp,'String', handles.mob_intersect);
else
    set(handles.r3 Disp,'String', handles.intersection(3));
    set(handles.r2 Disp,'String', handles.intersection(2));
    set(handles.r1 Disp,'String', handles.intersection(1));
    set(handles.mobile Disp,'String', handles.mob_intersect);
end

% Update handles structure
guidata(hObject, handles);

function r2_disp_Callback(hObject, eventdata, handles)
% hObject handle to r2_disp    (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Hints: get(hObject, 'String') returns contents of r2_disp as text
str2double(get(hObject, 'String')) returns contents of r2_disp
as a double

--- Executes during object creation, after setting all properties.
function r2_disp_CreateFcn(hObject, eventdata, handles)
% hObject handle to r2_disp (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function r3_disp_Callback(hObject, eventdata, handles)
% hObject handle to r3_disp (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of r3_disp as text
str2double(get(hObject,'String')) returns contents of r3_disp
as a double

--- Executes during object creation, after setting all properties.
function r3_disp_CreateFcn(hObject, eventdata, handles)
% hObject handle to r3_disp (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in clear_push.
function clear_push_Callback(hObject, eventdata, handles)
    % hObject    handle to clear_push (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles   structure with handles and user data (see GUIDATA)
    global node t_x t_y ob_x ob_y start count...
    mob_x mob_y txt pos J trail save_count;
    % resets ALL parameters inside the environment
    cla('reset');
    set(handles.r1_disp,'String','');
    set(handles.r2_disp,'String','');
    set(handles.r3_disp,'String','');
    set(handles.mobile_disp,'String','');
    set(handles.receiver_edit,'String',0);
    set(handles.ob_edit,'String',0);
    set(handles.R_edit,'String',1);
    set(handles.Q_edit,'String',1);
    set(handles.cellszize_edit,'String',10);
    set(handles.grid_check,'Value',0);
    set(handles.PLE_check,'Value',0);
    set(handles.crossPLE_check,'Value',0);
    set(handles.RSSI_check,'Value',0);
    handles.intersection = [];
    handles.mob_intersect = 0;
    handles.sizeofCell = 0;
    save_count = 0;
    handles.x = []; handles.y = [];
    handles.found_obj = [];
    handles.found_rec = [];
    handles.data = [];
    handles.data.node = [];
    handles.data.rssi = [];
    axis([0 100 0 100]);
    node = 0; t_y = 0; t_x = 0; ob_x = 0; ob_y = 0; start = 1;
    count = 1; mob_x = 0; mob_y = 0;
    txt = 1; pos = [];
    ghost_trail = findobj('Type','line','Color',[.7 .7 .9]);
    delete(ghost_trail);
    guidata(hObject, handles);

function ob_edit_Callback(hObject, eventdata, handles)
    % hObject    handle to ob_edit (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles   structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of ob_edit as text
% str2double(get(hObject,'String')) returns contents of ob_edit as a double

% --- Executes during object creation, after setting all properties.
function ob_edit_CreateFcn(hObject, eventdata, handles)
    hObject handle to ob_edit (see GCBO)
    eventdata reserved - to be defined in a future version of MATLAB
    handles empty - handles not created until after all CreateFcns called

    % Hint: edit controls usually have a white background on Windows.
    % See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),...
         get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function mobile_disp_Callback(hObject, eventdata, handles)
% hObject handle to mobile_disp (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of mobile_disp as text
% str2double(get(hObject,'String')) returns contents of mobile_disp as a double

% --- Executes during object creation, after setting all properties.
function mobile_disp_CreateFcn(hObject, eventdata, handles)
    hObject handle to mobile_disp (see GCBO)
    eventdata reserved - to be defined in a future version of MATLAB
    handles empty - handles not created until after all CreateFcns called

    % Hint: edit controls usually have a white background on Windows.
    % See ISPC and COMPUTER.
    if ispc && isequal(get(hObject,'BackgroundColor'),...
         get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
function cellsize_edit_Callback(hObject, eventdata, handles)
% hObject handle to cellsize_edit (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of cellsize_edit as
% text
% str2double(get(hObject,'String')) returns contents of
% cellsize_edit as a double

% --- Executes during object creation, after setting all properties.
function cellsize_edit_CreateFcn(hObject, eventdata, handles)
% hObject handle to cellsize_edit (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in crossPLE_check.
function crossPLE_check_Callback(hObject, eventdata, handles)
% hObject handle to crossPLE_check (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of crossPLE_check
global node t_x t_y mob_x mob_y crossPLE;

on = get(handles.crossPLE_check,'Value');
if on == 1
    obstacles = length(handles.found_obs);
    num_rec = length(node(:,1));
    mobile = [mob_x mob_y];
    % mobile to receiver node check
    for i = 1:num_rec
        for k = 1:obstacles
yes_no(k) = check_intersect(node(i,1), mob_x,...
handles.x(2*k-1), handles.x(2*k),node(i,2),...
mob_y, handles.y(2*k-1), handles.y(2*k));
end

reality = sum(yes_no);
if reality >= 1;
    handles.data.node(i) = 2.5;
else
    handles.data.node(i) = 2.1;
end
end

%# labels correspond to their order
for i  = l:num_rec
    labels(i) = cellstr( num2str(handles.data.node(i)) );
    text(((node(i,l)+mob_x)/2+l),  ((node(i,2)+mob_y)/2+l),...
        labels(i), 'VerticalAlignment','bottom',...
'HorizontalAlignment','right','Color',...
[0,1,.9],'FontSize',8);
    line([node(i,1) mob_x],[node(i,2) mob_y],...
    'LineStyle', ' ', 'Color',[0,1,.9],'LineWidth','.8);
end
crossPLE = max(mean(handles.data.node))
else
    test = findobj('Color',[0,1,.9]);
delete(test);
%crossPLE = 0;
labels = [];
end
guidata(hObject, handles);

% --- Executes on button press in RSSI_check.
function RSSI_check_Callback(hObject, eventdata, handles)
% hObject handle to RSSI_check (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of RSSI_check
global pos t_x t_y mob_x mob_y node step;
on = get(handles.RSSI_check,'Value');
if on == 1
noise = 10.*rand(1,1);
if noise > 5
    noise = noise - 5;
    noise = -noise;
end
noise = ceil(noise);
%introduce random noise into RSSI because of random nature

[t1,t2] = size(pos);
obstacles = length(handles.found_obs);
for i = 1:t1
    for j = 1:t1
        for k = 1:obstacles
            yes_no(k) = check_intersect(pos(i,1), t_x,...
                                        handles.x(2*k-1), handles.x(2*k),pos(j,2), t_y,...
                                        handles.y(2*k-1), handles.y(2*k));
        end
        reality = sum(yes_no);
        dist = getDistance(pos(i,1), pos(j,2), t_x, t_y)/step;
        %reduces distance to step size, therefore making each unit
        %distance one step in space
        x = log10(dist);

        if reality >= 1;
            handles.data.rssi(i,j) = ceil(-24.904*x + 1) + 38 +...
                                        noise;
        else
            handles.data.rssi(i,j) = ceil(-20.995*x + .8946) +...
                                        45 + noise;
        end
    end
end

%# labels correspond to their order
for i = 1:t1
    for j = 1:t1
        labels(i,j) = cellstr( num2str(handles.data.rssi(i,j)));
        text(pos(i,1), pos(j,2), labels(i,j),...
             'VerticalAlignment','bottom',...
             'HorizontalAlignment','right','Color',...
             [.9 0 0],'FontSize',6);
    end
end
else
    test = findobj('Color', [.9 0 0]);
    delete(test);
end
guidata(hObject, handles);

% --- Executes on button press in adapt_push.
function adapt_push_Callback(hObject, eventdata, handles)
    % hObject    handle to adapt_push (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    global trail pos t_x t_y mob_x mob_y node J step finish;

    finish = false;

    %while(~finish)
    mobileNode = findobj(gca, 'Type', 'line', '-and', 'Color', 'c');
    ghost = findobj(gca, 'Type', 'text', '-and', 'Color', [0 0 .2]);
    obstacles = length(handles.found_obs);
    x_pos = get(mobileNode, 'Xdata');
    y_pos = get(mobileNode, 'Ydata');
    Q = str2double(get(handles.Q_edit, 'String'));
    R = str2double(get(handles.R_edit, 'String'));

    mobilePos = [x_pos y_pos];
    mobilePos1 = [x_pos (y_pos+step)];
    mobilePos2 = [(x_pos+step) y_pos];
    mobilePos3 = [x_pos (y_pos-step)];
    mobilePos4 = [(x_pos-step) y_pos];

    position = ceil(mobilePos./step);

    first = position(1);
    second = position(2);

    cross_PLEold = cross_ple(node, mobilePos, obstacles,...
        handles.x, handles.y);
    cross_PLE(1) = cross_ple(node, mobilePos1, obstacles,...
        handles.x, handles.y);
    cross_PLE(2) = cross_ple(node, mobilePos2, obstacles,...
        handles.x, handles.y);
    cross_PLE(3) = cross_ple(node, mobilePos3, obstacles,...
handles.x, handles.y;
cross_PLE(4) = cross_ple(node, mobilePos4, obstacles,...
handle.x, handle.y);

J_old = Q*(1 / handles.data.rssi(first,second)) + R*cross_PLEold;
J(1) = Q*(1 / handles.data.rssi(first,second+1)) + R*cross_PLE(1); %up
J(2) = Q*(1 / handles.data.rssi(first+1,second)) + R*cross_PLE(2); %right
J(3) = Q*(1 / handles.data.rssi(first,second-1)) + R*cross_PLE(3); %down
J(4) = Q*(1 / handles.data.rssi(first-1,second)) + R*cross_PLE(4); %left

[minVal, minIndex] = min(J);
done = false;

%Obstacle avoidance with adaptive behavior

while(¬done)
    switch minIndex
        case 1
            move = 'up';
            for k = 1:obstacles
                yes_no(k) = check_intersect(pos(first,1),...
                    pos(first,1), handles.x(2*k-1), handles.x(2*k),...
                    pos(second,2), pos(second+1,2),...
                    handles.y(2*k-1), handles.y(2*k));
            end
            reality = sum(yes_no);
            if reality >= 1;
                if J(2) < J(4) %plans to go right
                    move = 'right';
                    test = moveMobile(mobilePos(1), mobilePos(2),...
                        move, step);
                    if test == handles.previous %prevent loop, force left
                        done = true;
                        move = 'left';
                        handles.next = moveMobile(mobilePos(1),...
                            mobilePos(2), move, step);
                        break
                    end
                else %go right as planned
                    minIndex = 2;
                end
            else %plans to go left
                move = 'left';
            end
        end
    end
end
test = moveMobile(mobilePos(1), mobilePos(2), move, step);
if test == handles.previous %prevent loop, go right
    done = true;
    move = 'right';
    handles.next = moveMobile(mobilePos(1),
        mobilePos(2), move, step);
    break
else %go left as planned
    minIndex = 4;
end
else
    done = true;
    handles.next = moveMobile(mobilePos(1), mobilePos(2),
        move, step);
end

case 2
move = 'right';
for k = 1:obstacles
    yes_no(k) = check_intersect(pos(first,1), ...
        pos(first+1,1), handles.x(2*k-1),...
        handles.x(2*k), pos(second,2), pos(second,2),...
        handles.y(2*k-1), handles.y(2*k));
end
reality = sum(yes_no);
if reality >= 1;
    if J(1) < J(3)
        move = 'up'; %plans to move up
        test = moveMobile(mobilePos(1), mobilePos(2),
            move, step);
        if test == handles.previous %prevent loop, force down
            done = true;
            move = 'down';
            handles.next = moveMobile(mobilePos(1),
                mobilePos(2), move, step);
            break
        else %go up as planned
            minIndex = 1;
    end
end
else % plans to go down
    move = 'down';
    test = moveMobile(mobilePos(1), mobilePos(2), ... 
        move, step);
    if test == handles.previous % prevent loop, go up
        done = true;
        move = 'up';
        handles.next = moveMobile(mobilePos(1), ... 
            mobilePos(2), move, step);
    break

else % go down as planned
    minIndex = 3;
    end
end
else
    done = true;
    handles.next = moveMobile(mobilePos(1), mobilePos(2), ... 
        move, step);
end

case 3
    move = 'down';
    for k = 1:obstacles
        yes_no(k) = check_intersects(pos(first,1), ...
            pos(first,1), handles.x(2*k-1), handles.x(2*k), ...
            pos(second,2), pos(second-1,2), ...
            handles.y(2*k-1), handles.y(2*k));
    end
    reality = sum(yes_no);

    if reality >= 1;
        if J_2 < J_4
            move = 'right';
            test = moveMobile(mobilePos(1), mobilePos(2), ... 
                move, step);
            if test == handles.previous % prevent loop, force left
                done = true;
                move = 'left';
                handles.next = moveMobile(mobilePos(1), ... 
                    mobilePos(2), move, step);
            break
        else % go right as planned
            minIndex = 2;
        end
    end
end
end
else  %plans to go left
    move = 'left';
    test = moveMobile(mobilePos(1), mobilePos(2),
                     move, step);
    if test == handles.previous %prevent loop, go right
        done = true;
        move = 'right';
        handles.next = moveMobile(mobilePos(1),
                                   mobilePos(2), move, step);
        break
    end
else %go left as planned
    minIndex = 4;
end
end
else
    done = true;
    handles.next = moveMobile(mobilePos(1), mobilePos(2),
                               move, step);
end

case 4
move = 'left';
for k = 1:obstacles
    yes_no(k) = check_intersect(pos(first,1),
                                pos(first-1,1), handles.x(2*k-1), ...
                                handles.x(2*k),pos(second,2), pos(second,2),...
                                handles.y(2*k-1), handles.y(2*k));
end
reality = sum(yes_no);

if reality >= 1;
    if J(1) < J(3)
        move = 'up'; %plans to move up
        test = moveMobile(mobilePos(1), mobilePos(2),
                           move, step);
        if test == handles.previous %prevent loop, force down
            done = true;
            move = 'down';
            handles.next = moveMobile(mobilePos(1),
                                      mobilePos(2), move, step);
            break
        end
    end
    else %go up as planned

minIndex = 1;

end
else %plans to go down
move = 'down';
test = moveMobile(mobilePos(1), mobilePos(2),
    move, step);
if test == handles.previous %prevent loop, go up
done = true;
move = 'up';
handles.next = moveMobile(mobilePos(1),
    mobilePos(2), move, step);
break
else %go down as planned
    minIndex = 3;
end
end
else
done = true;
handles.next = moveMobile(mobilePos(1), mobilePos(2),
    move, step);
end
end

dis_from_tgt = getDistance(t_x, t_y, mobilePos(1), mobilePos(2));
handles.distance = [handles.distance; dis_from_tgt];

if minVal <= J_old
    cont = true;
    if dis_from_tgt < step
        cont = false;
    end
else
    cont = false;
end

delete(mobileNode);
delete(ghost);
%initial movement

handles.previous = [mobilePos(1) mobilePos(2)];
trail = [trail; mobilePos];

if cont == false
    disp('simulation finished');
    handles.next = mobilePos;
    finish = true;
end

%check for obstacle--change path if necessary

trail = [trail; handles.next];
mob_x = handles.next(1);
mob_y = handles.next(2);
plot(handles.next(1),handles.next(2),'c.','MarkerSize',35);
plot(trail(:,1),trail(:,2),'Color',[.7 .7 .9],'LineWidth',.6,...
    'LineStyle',':');
text(handles.next(1),handles.next(2), 'M', 'VerticalAlignment',...
    'bottom','HorizontalAlignment', 'right','Color', [0 0 .2]);

guidata(hObject, handles);

function R_edit_Callback(hObject, eventdata, handles)
    % hObject    handle to R_edit (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Hints: get(hObject,'String') returns contents of R_edit as text
    % str2double(get(hObject,'String')) returns contents of R_edit
    % as a double
    r = get(handles.R_edit, 'String');
    rval = str2double(r);
    if rval < 0
        rval = 0;
        set(handles.R_edit, 'String',rval);
    elseif rval > 1000
        rval = 1000;
        set(handles.R_edit, 'String',rval);
    end
    %qval = 1000 - rval;
    %set(handles.Q_edit,'String',qval);
    guidata(hObject, handles);
function R_edit_CreateFcn(hObject, eventdata, handles)
% hObject handle to R_edit (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Q_edit_Callback(hObject, eventdata, handles)
% hObject handle to Q_edit (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Q_edit as text
% str2double(get(hObject,'String')) returns contents of Q_edit as a double
q = get(handles.Q_edit, 'String');
qval = str2double(q);
if qval < 0
    qval = 0;
    set(handles.Q_edit,'String',qval);
elseif qval > 1000
    qval = 1000;
    set(handles.R_edit,'String',qval);
end
guidata(hObject, handles);

function Q_edit_CreateFcn(hObject, eventdata, handles)
% hObject handle to Q_edit (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in clr_trail_push.
function clr_trail_push_Callback(hObject, eventdata, handles)
% hObject    handle to clr_trail_push (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global trail
ghost_trail = findobj('Type','line','Color',[.7 .7 .9]);
delete(h_ghost_trail);
trail = []; guidata(hObject, handles);

guidata(hObject, handles);

% --- Executes on button press in save_push.
function save_push_Callback(hObject, eventdata, handles)
% hObject    handle to save_push (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global save_count trail node t_x t_y;

save_count = save_count + 1;
filename = strcat('PLE_data', num2str(save_count));
distance = handles.distance;
path = trail;
Q = str2double(get(handles.Q_edit,'String'));
R = str2double(get(handles.R_edit,'String'));
static = node;
target = [t_x t_y];
box = [handles.x; handles.y];
save(fullfile...
    ('C:\Documents and Settings\Miguel\My Documents\MATLAB\Results',...     filename), 'distance', 'path', 'Q', 'R', 'static', 'target', 'box');
handles.distance = [];
trail = [];
guidata(hObject, handles);
BIBLIOGRAPHY


Of-The-Earth UAV Flight in A Constrained Urban Environment,” Proc. 28th

and Evaluating MAV Control Algorithms in a Realistic Urban Setting,” 2005

[76] Aranda, S., Martínez, S., and Bullo, F., “On Optimal Sensor Placement and
Motion Coordination for Target Tracking,” Proc. 2005 IEEE International


[78] Mostofi, Y., “Compressive Cooperative Sensing and Mapping in Mobile Networks,”


Institute of Aeronautics and Astronautics (2005).

Aware Unmanned Aircraft System,” AIAA Journal of Guidance, Control,
and Dynamics (2011).

[82] Corbets, J. and Langelaan, J. “Parameterized Trajectories for Target Localization
Using Small and Micro Unmanned Aerial Vehicles,” Proc. of the American Control
