University of Nevada, Reno

An Extended Potential Field Controller for use on Aerial Robots

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

by

Alexander C Woods

Dr. Hung M. La - Thesis Advisor
May 2016

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

ALEXANDER C WOODS

Entitled

An Extended Potential Field Controller For Use On Aerial Robots

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Hung M. La, Ph.D., Advisor

Henry Fu, Ph.D., Committee Member

Eric Wang, Ph.D., Committee Member

Monica Nicolescu, Ph.D., Graduate School Representative

David W. Zeh, Ph. D., Dean, Graduate School

May, 2016

Abstract

This thesis focuses on the design and implementation of an extended potential field controller (ePFC) which enables a quadcopter aerial robot to track a dynamic target while simultaneously avoiding obstacles in the environment. The design of the ePFC extends the foundational concepts of a traditional potential field controller (PFC), which uses attractive and repulsive potential fields to navigate toward a target and avoid obstacles. A traditional PFC is a function of only the relative positions of the drone to the target and obstacles, respectively, and has shortcomings for aerial robots which are much harder to control than ground robots. The proposed ePFC takes into account the relative velocities of the drone to the target and obstacles, respectively, in addition to the relative positions which enhances the controller's ability and improves performance. The proposed controller is simulated using Matlab's Simulink tool, and the simulation results show that the ePFC reduces the overshoot of the robot's location in response to a step input by 19% and the settling time by nearly 17% when compared to a traditional PFC. The proposed controller is implemented on an experimental platform, the ARDrone 2.0, and the obtained results show that the drone is able to track both static and dynamic targets, moving in either set or arbitrary patterns, all while avoiding obstacles in the test space. Compared to the simulation, the experimental results show an overshoot 2% higher, and a settling time only 0.5 sec slower.

Acknowledgments

I would like to thank my advisor, Dr. Hung La, who provided the support, encouragement, and insights to keep me on the right path.

Additionally, I would like to thank Dr. Kam Leang, who inspired me to pursue graduate studies. I enjoyed the time I was able to spend working with Kam, and am lucky to have had his guidance during the my initial time at UNR.

To my committee members, Dr. Eric Wang and Dr. Henry Fu, thank you for the time and effort which you have put in to review this thesis and for providing advice along the way.

Finally I would like to thank my amazing wife, Amanda, who has patiently handled several tough decisions and who has always offered her support and love when it was needed most. I could not have done this without her.

Dedication

Dedicated to Grandpa Ron.

Table of Contents

1 Thesis Introduction, Contribution, and Organization **1**

1.1 Introduction 1

1.2 Contribution 2

1.3 Organization 2

2 Background **4**

2.1 Unmanned Autonomous Systems (UAS) 4

2.1.1 Fixed Wing 5

2.1.2 Multirotors 6

2.2 UAS Sensing 7

2.2.1 Light Detection and Ranging 7

2.2.2 Computer Vision 9

2.2.3 Summary of Sensing Solutions 11

2.3 UAS Navigation 11

2.3.1 Deliberative Algorithms 12

2.3.2 Reactive Algorithms 14

2.3.3 Modern UAS Control 14

2.3.4 Summary of UAS Navigation 15

2.4 Summary 16

3 System Model **17**

3.1 Reference Frames 17

3.2 Euler Transformations 19

3.3 Newton-Euler Equations 20

3.4 Quadcopter Dynamics 23

3.5 Summary 26

4 Design of an ePFC **27**

4.1 Traditional Potential Field Controller 27

4.2 Extended Potential Field Controller 32

4.3 Stability Analysis 36

4.4 Summary 37

5 Simulation **38**

5.1 MATLAB Environment 38

5.2 Simulation results 39

5.3 Summary 42

6 Experimental Setup, Results, and Discussion **43**

6.1 Experimental Setup 43

6.2 Experimental Results and Discussion 47

6.3 Summary 54

7 Conclusions and Future Work **55**

7.1 Conclusions 55

7.2 Future Work 55

List of Tables

2.1 Dijkstra's Algorithm Progression 12

5.1 Simulation Waypoints 39

5.2 Simulation Controller Evaluation 41

6.1 Experimental Tests 47

6.2 Experimental Waypoints 50

6.3 Simulation vs Experimental Evaluation 52

List of Figures

1.1 Using an extended potential field controller, an aerial robot can safely navigate to a target location while avoiding obstacles in its path. . . . 3

2.1 A fixed wing UAS produces lift by utilizing airflow over fixed airfoils, and uses one or more engines to propel itself forward. 5

2.2 Multirotor UAS, such as the quadcopter shown here, produce lift by directly actuating propellers. Lateral movement is achieved through pitching or rolling the aircraft to change the direction of the thrust vector. 6

2.3 LIDAR provides a 2D planar sweep of distance data with a very wide field of view. However, objects above or below the plane cannot be sensed. 8

2.4 By holding lateral position constant while performing an ascent, multiple LIDAR scans can be combined with altitude information to form a 3D representation of an environment. In this figure, a point cloud of the environment shown in Fig. 2.3 was created using the technique described. 9

2.5 (a) The Microsoft Kinect is a very useful tool for localization because it provides depth information by emitting a (b) grid of infrared dots and then taking an image in order to reconstruct the environment. . . 10

2.6 In Dijkstra's algorithm, the environment is represented as a graph of nodes connected by weighted edges. The weight can represent any cost, but is most typically associated with distance. In this particular case, the shortest path from node A to node E is highlighted in red. 13

3.1 An inertial reference frame, E, is fixed on earth, and an accelerating body reference frame, B, is fixed at the rigid bodie's center of mass. The position and velocity of the rigid body are designated as $\vec{p}$ and $\vec{v}$ respectively. Euler rotation matrices may be used to transform between frames E and B for a given orientation. 18

3.2 A quadcopter is symmetric about both the $\hat{b}_x$ and $\hat{b}_y$ axis. Each motor produces a force, $\vec{T}_i$ and moment, $\vec{M}_i$, which are functions of the motor's angular velocity, Ω_i. 23

4.1 An example of a traditional potential field which can be used for navigating toward a target while avoiding multiple obstacles. 28

4.2 The repulsive force due to an obstacle should be zero when the drone is sufficiently far from it. In this case, the limit is P^*. While outside of this limit, the drone is unaffected by the obstacle's repulsive potential. However, as soon as the relative distance becomes less than P^*, the drone takes avoidance action. 31

4.3 By taking into account the time rate of change of $\|\vec{p}_{do}\|$, the controller is able to ignore repulsive effects from the obstacle if the drone is moving away from the obstacle ($\|\dot{p_{do}}\| \geq 0$). This results in a more direct route to the target, thus saving time. 35

5.1 The Simulink model used includes a state space representation from the ARDrone Simulink Development Kit, as well as custom blocks for the ePFC controller described in this work. 39

5.2 A traditional PFC is simulated on the ARDrone, with poor results. Because drones cannot stop instantaneously like ground robots, the drone often overshoots the desired waypoint. For reference, the drone takes approximately 35 seconds to complete a full loop of the course. 40

5.3 Using the extended potential field controller (ePFC), the drone is able to complete the course without overshooting the target waypoints, and avoids the obstacle by a larger margin than the traditional controller. Because the drone does not overshoot the target, it is able to complete the course in a shorter amount of time compared to the traditional controller. 40

5.4 A more complex simulation was performed with multiple obstacles placed throughout the environment. The drone is able to successfully navigate between waypoints without colliding with a single obstacle, thus demonstrating its effectiveness in multi-obstacle scenarios. 41

6.1 A low cost, commercially available quadrotor drone is used as the experimental platform for testing and demonstrating the effectiveness of the proposed control method. 44

6.2 The Motion Analysis Cortex software gives the user a real-time, visual 3D representation of the environment including camera locations and any objects sensed by the system [1]. 45

6.3 A Matlab GUI was created to show the positions of the drone, target, and any obstacles present. It also allows the user to control when the drone takes off, lands, tracks the target, or simply hovers in place. . . 45

6.4 The experimental setup for this work includes a Motion Analysis external tracking system with 16 cameras which provides the position of the drone and obstacles in the environment. The same Simulink model used in Chapter 5 provides control commands to the ARDrone over a wireless connection. 46

6.5 The results of tracking a static target with no obstacles are very good. With an initial condition of approximately 3.2m, the drone achieves position in under 5sec. 48

6.6 The results of tracking a static target with an obstacle in the way demonstrates the controllers effectiveness at avoiding collisions. As expected, the drone's position relative to the obstacle decreases, but the drone takes avoidance action and never gets closer than one meter away from the obstacle. 49

6.7 In the third experiment, the drone tracks a target which moves in an approximate rectangle. As shown, the drone does track, but because the desired trajectory is human-controlled the reference is not perfect. Test number four addresses this imperfection by using set virtual waypoints. 49

6.8 To test the ePFC experimentally, the drone follows waypoints similar to those in the simulation. The drone successfully reaches each waypoint and avoids the obstacle in its path between waypoints two and three. 50

6.9 Stills from a video demonstrate the drone taking avoidance action while tracking moving waypoints. 51

6.10 The error in response to a waypoint change, or a step input, results in a settling time of approximately 5.5sec and a percent overshoot of only 1.8%. The X axis was chosen because the step input for this direction was the largest, at over 2.5m, whereas the Y input is only 1m. 52

6.11 The third experiment demonstrates the drone tracking a dynamic target, while maintaining the proper heading to always face the target. . 53

Chapter 1

Thesis Introduction, Contribution, and Organization

1.1 Introduction

Recent advances in the field of unmanned autonomous systems (UAS) have drastically increased the potential uses of both unmanned ground vehicles (UGV) and unmanned aerial vehicles (UAV). UAS can be utilized in situations which may be hazardous to human operators, such as assisting wild land fire fighters [2–6], search and rescue in hazardous conditions or locations [7–11], and disaster remediation [12–14]. Additionally, UAS can be used in repetitive or tedious work where a human operator may lose focus such as infrastructure inspection [15–17], agricultural inspections [18, 19], and environmental sensing [20]. By using UAS in these repetitive applications, accuracy and precision can be maintained where a human operator might make an error.

Interest in the UAS industry has grown rapidly, but practical applications are hindered by challenges that limit their usefulness for real world applications. For example, UAS rely heavily on a global positioning system (GPS) for self localization. This works well in many situations, but severely limits their use in GPS-denied environments. Additionally, most UAS have limited sensing capabilities for real-time obstacle detection. Aside from sensing specific challenges, UAS also require advanced mapping, planning, and navigation algorithms to safely navigate and interact with their surroundings.

This thesis presents an extended potential field controller (ePFC) which enables an aerial robot to safely navigate to a target location while simultaneously avoiding obstacles in its path as demonstrated in Fig. 1.1. This thesis presents the design, simulation, and implementation of the novel ePFC on an ARDrone 2.0 quadcopter in a laboratory environment with an external motion capture system [1].

1.2 Contribution

The contribution of this work is the design, simulation, and experimental implementation of an extended potential field controller (ePFC) for navigation of an aerial robot. The extended potential field controller presented enables the robot to not only smoothly navigate towards a target position but to also avoid both static and dynamic obstacles in its path. The ability to safely navigate in a dynamic environment is critical for the implementation of robots in real-world scenarios as outline in Section 1.1.

1.3 Organization

This thesis is organized as follows. Chapter 2 presents a brief introduction to unmanned autonomous systems, sensing systems, and control methods. Chapter 3

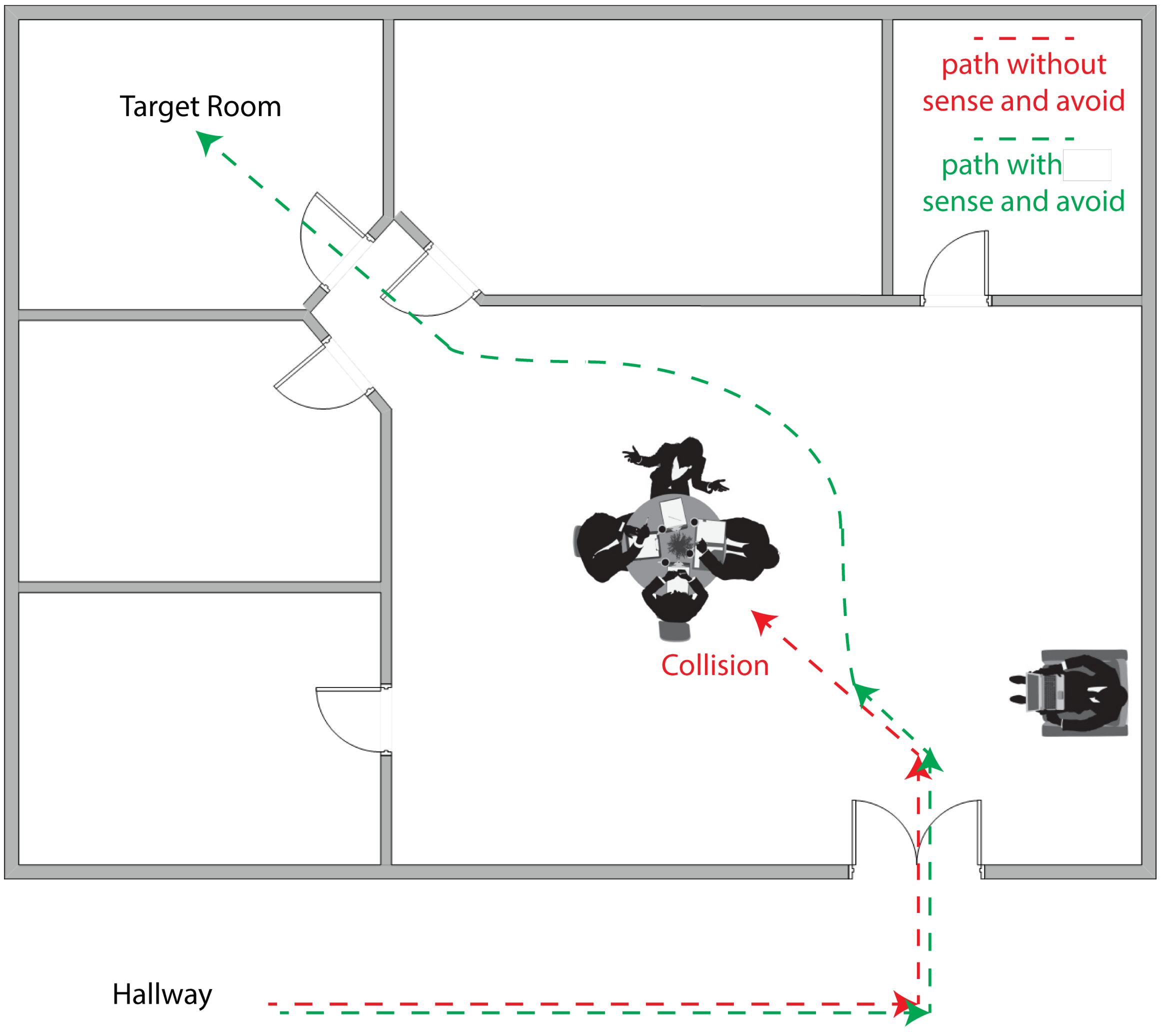


Figure 1.1: Using an extended potential field controller, an aerial robot can safely navigate to a target location while avoiding obstacles in its path.

presents the development of the quadcopter system dynamics, which is used as the experimental platform for this thesis. Chapter 4 presents the mathematical design and development of the extended potential field controller. Chapter 5 presents simulations based on the developed model, system characteristics, and controller. Chapter 6 presents experimental results and discussion. Finally, Chapter 7 discusses thoughts on future work and concludes the thesis.

Chapter 2

Background

2.1 Unmanned Autonomous Systems (UAS)

Unmanned autonomous systems (UAS) are robotic systems which can perform tasks autonomously without human input. Because there are no human operators or passengers on-board, UAS can perform tasks in areas that may be dangerous (e.g., extreme environmental conditions, areas affected by chemical weapons, in enemy territory, or radioactive fallout regions). In addition to uses where preservation of human life is paramount, UAS also have great potential for tedious and repetitive tasks during which a human operator is prone to error such as industrial, agricultural, or infrastructure inspections. Because of their wide range of applications, and the ability to perform tasks autonomously, UAS have the potential to make a very positive impact on our society. In the field of aerial UAS, there are two general categories of platform: fixed wing and multirotors.

2.1.1 Fixed Wing

Fixed wing UAS, such as the one shown in Fig. 2.1, generally operate by employing airfoils as the primary mechanism for lift and utilize a separate motor to propel the aircraft forward. Because this method of flight is very efficient, fixed wing UAS are generally well suited to missions where endurance is critical. For example, fixed wing UAS excel in tasks such as fire monitoring, target tracking and surveillance, convoy protection, atmospheric sampling, and general environmental monitoring - all tasks which require extended flight times.

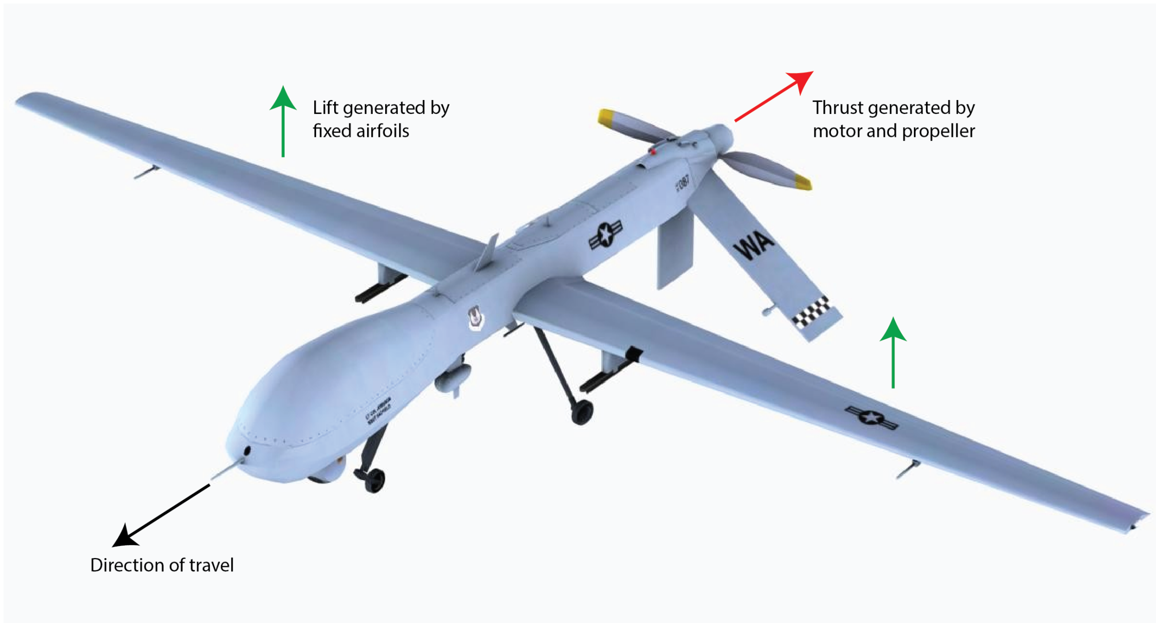

Lift generated by fixed airfoils
Thrust generated by motor and propeller
Direction of travel

Figure 2.1: A fixed wing UAS produces lift by utilizing airflow over fixed airfoils, and uses one or more engines to propel itself forward.

While efficient, fixed wing aircraft must always maintain forward movement in order to generate lift and cannot hover in place. This is generally an acceptable compromise for most missions where orbiting a target location at high altitude is acceptable, but it does place limitations on the uses of fixed wing UAS. For example, fixed wing UAS cannot operated in environments which require high levels of agility, such as urban environments, forested areas, or indoors.

2.1.2 Multirotors

In contrast to fixed wing UAS, multirotor UAS use multiple rotors oriented vertically in the body frame to produce lift. Most multirotors are equipped with at least three rotors, in which case one of the rotors must be equipped with an additional actuator, and may have upwards of eight rotors. However, one of the more popular configurations is the quadcopter, which utilizes four rotors as shown in Fig. 2.2 - one pair rotating clockwise and the other rotating counter clockwise. The quadcopter is an underactuated system, which has only four actuators and six degrees of freedom. This leads to two of the degrees of freedom being dependent upon each other. In order to move forward or backward, the quadcopter must change its pitch. Similarly, in order to move left or right, the quadcopter adjusts its roll. The remaining degrees of freedom are independent. The quadcopter can change its heading by utilizing the moment produced by the rotating motors - increasing the angular velocity of one pair, and decreasing the other. The altitude of the platform can be controlled by increasing or decreasing the total thrust produced by the motors.

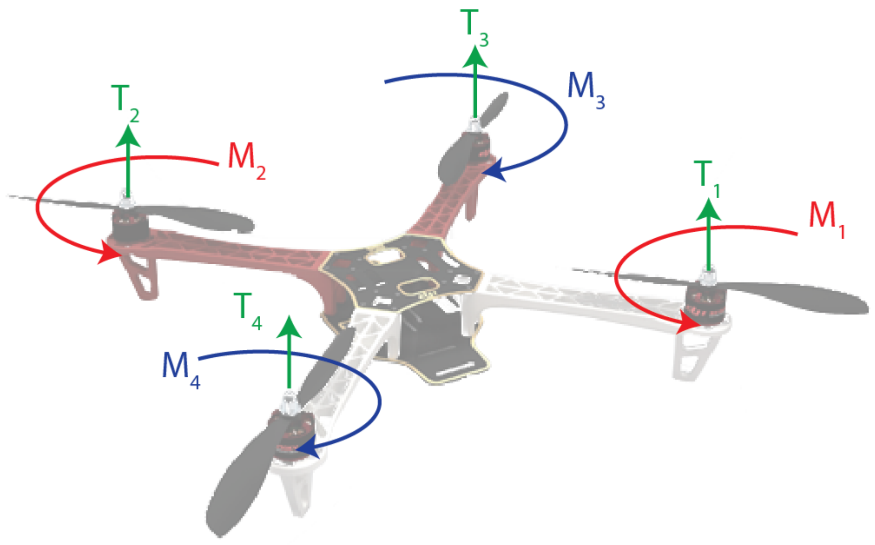


Figure 2.2: Multirotor UAS, such as the quadcopter shown here, produce lift by directly actuating propellers. Lateral movement is achieved through pitching or rolling the aircraft to change the direction of the thrust vector.

Because multirotor UAS do not depend on forward motion for lift, they are better equipped than fixed wing UAS to handle situations which require vertical take off and landing, high levels of agility, or the ability to hover in place. These capabilities are very desirable in environments such as indoors, forested areas, urban environments, or densely populated areas. However, these capabilities come at a cost - efficiency. Multirotor UAS are not capable of the flight times that fixed wings UAS are.

2.2 UAS Sensing

While UAS have many real-world applications, current technology relies heavily upon GPS for self-localization. This poses a problem for UAS operating in GPS-denied environments such as in buildings, canyons, forested areas, and many urban environments. In addition to self-localization challenges, UAS also face challenges locating potential obstacles in their surrounding environment. Because of the lack of robust sensing schemes, much of the research in the UAS field has focused on improving sensing and obstacle avoidance capabilities.

2.2.1 Light Detection and Ranging

One of the several promising fields of on-board sensing methods is light detection and ranging (LIDAR). Most commercial LIDAR products provide a 2D planar sweep of distance data with a very wide field of view (e.g., 270°) as shown in Fig. 2.3.

LIDAR is effective at sensing objects that lie within the sensing plane, but is otherwise blind. Because of this limitation, alternate strategies must be employed to get a full 3D view of the environment. For example, the robot can perform an ascent while keeping its lateral position constant and stitch together horizontal sweeps to get a full 3D representation of the environment in its field of view, as shown in Fig. 2.4.

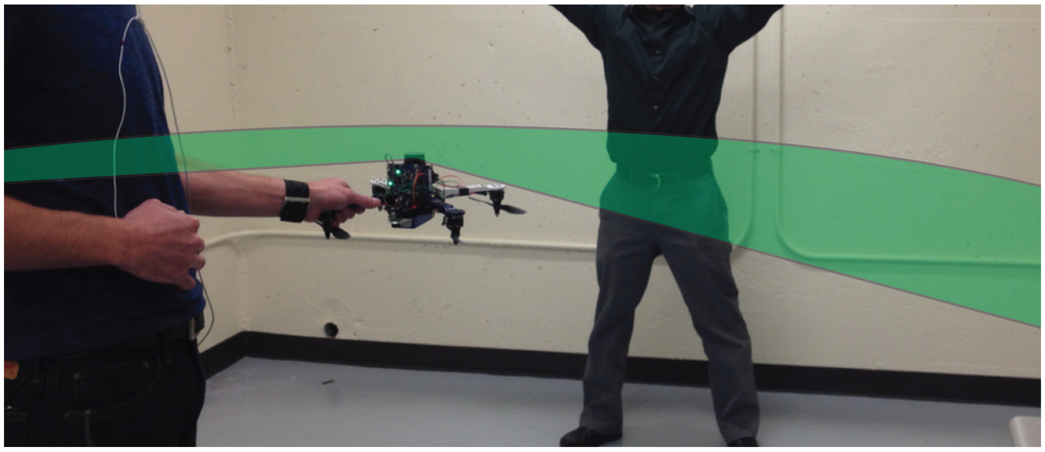


Figure 2.3: LIDAR provides a 2D planar sweep of distance data with a very wide field of view. However, objects above or below the plane cannot be sensed.

There are several commercially available LIDAR units, which are quite accurate ($\pm$50)mm and can provide range measurements up to 30m [21]. Independent studies have found that these LIDAR units often perform better than their specifications for most surface properties, but that shiny surfaces can produce poor results. One such study found that maximal errors were approximately 140mm for shiny surfaces, 30mm for matte surfaces, and 32mm for gray surfaces [22]. These results indicate that LIDAR would be a very useful sensor for robot navigation. However, sensing obstacles with shiny surfaces at incidence angles greater than $\pm 10^\circ$ often results in bad readings from the sensor, requiring the robot to be closer to the obstacle before sensing it reliably, thus reducing the effective range of the sensor.

Despite this limitation, several groups have successfully investigated the use of LIDAR as a means of localization. One group employed a reflexive algorithm in combination with a LIDAR sensor for simulating navigation through an unknown environment [23]. Another group developed a multilevel simultaneous localization and mapping (SLAM) algorithm which utilized LIDAR as its primary sensing method [24]. Other groups have used LIDAR on autonomous vehicles for control and multi-floor navigation [25, 26].

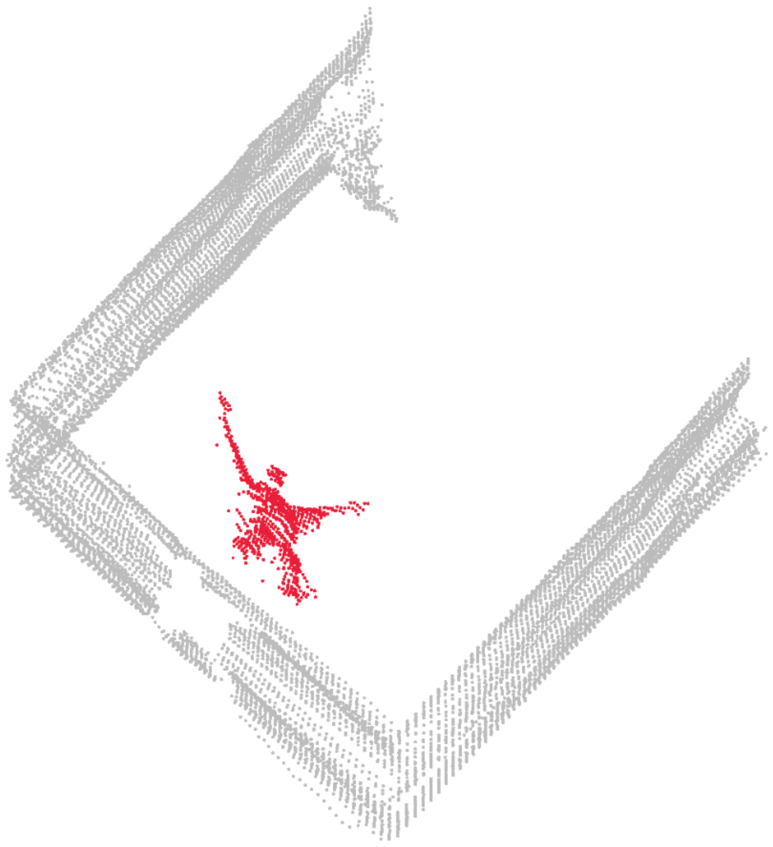


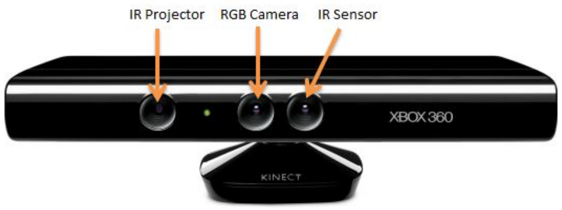
Figure 2.4: By holding lateral position constant while performing an ascent, multiple LIDAR scans can be combined with altitude information to form a 3D representation of an environment. In this figure, a point cloud of the environment shown in Fig. 2.3 was created using the technique described.

2.2.2 Computer Vision

Another major area of research for localization in GPS-denied environments is computer vision, which utilizes cameras images and complex algorithms to gather data about the environment. Many unmanned systems are already equipped with cameras which are used for information gathering purposes, and as a means of manually flying without visual line of sight from a first person view (FPV). Since they are often already equipped, cameras present a unique opportunity to minimize system weight if they can be used for localization as well.

In the field of computer vision, there are a variety of techniques that can be used to gather position and attitude information. One group used a single camera, looking at an object of known size to determined its location [27]. In this application, blob detection using an open source computer vision library, OpenCV, was implemented on an on-board Raspberry Pi computer. The position of the blob on the vertical axis gives the error in altitude, while the position on the horizontal axis gives the error in heading. To determine the range of the object, the authors develop a relationship between the square of the pixels (area) and the range. Naturally, a smaller area indicates a farther range and vice versa. Both a low pass filter and a Kalman filter are implemented to mitigate noisy raw measurements.

Another group utilized a combination of LIDAR and a Microsoft Kinect sensor to explore an unknown environment [28]. The Microsoft Kinect sensor is a very useful sensor for robotic exploration because it utilizes image-based 3D reconstruction to provide depth measurements. It does this by emitting a known pattern of infrared dots as shown in Fig.2.5b, and then uses images of the environment to calculate depths. Using depth information provided by both the LIDAR and Kinect sensor, this group was able to map and explore an unknown environment.



(a) Microsoft Kinect sensor

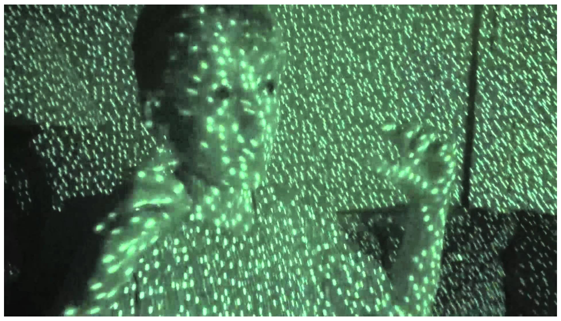

(b) Kinect sensor infrared dot pattern

Figure 2.5: (a) The Microsoft Kinect is a very useful tool for localization because it provides depth information by emitting a (b) grid of infrared dots and then taking an image in order to reconstruct the environment.

Several other groups have also successfully used computer vision as a means of localization [29, 30] and it is proving to be a very promising method of operating in GPS-denied environments.

2.2.3 Summary of Sensing Solutions

While only two potential methods of UAS sensing are presented here, many others exist or are being developed by research groups across the world. However, despite the efforts of the research community, sensing challenges have not been completely solved. This thesis does not attempt to solve the sensing challenges facing UAS, but instead focuses on navigational challenges.

2.3 UAS Navigation

Navigation methods can generally be broken down into deliberative (e.g., preplanned trajectory) and reactive (e.g., real-time control) subcategories. For example, a deliberative map based navigation algorithm may consider the given data and decide upon an optimal route to take. These types of navigation systems are generally quite computationally expensive and therefore may take a significant amount of time if performed on board a platform with limited processing power or if the map is fairly complex. Additionally, if the map data changes, such as the introduction of a new obstacle, the algorithm must re-plan the entire route. Therefore, these methods are ideal for highly predictable environments. In contrast, most reactive methods simply make the best decision they can in the moment with whatever information they have. This generally leads to faster decision making and makes them better suited to changing environments. However, this comes at the cost of optimization. Routes taken by reactive navigation methods are generally not as efficient as a deliberative method.

2.3.1 Deliberative Algorithms

Dijkstra's algorithm, published in 1959 by Edsger Dijkstra, is considered a foundation of deliberative, map based navigation methods. This algorithm works by utilizing a map which represents the environment as a graph of nodes with weighted edges such as that shown in Fig. 2.6. The weight can be determined by distance, or any other metric which a user would like to optimize. To begin, the starting node is given a weight of zero, and all other nodes are assigned weights of infinity as shown in step one in Table 2.1. Next, starting at the initial node, we evaluate any directly reachable node by using the weights of connecting edges. If the weight of the edge plus the weight of the current node (zero for the initial node) is less than the current value, then the weight is updated. The algorithm then moves on to a new node which must be unvisited, and is chosen by its weight. So, for example, moving from step 3 to step 4 of the algorithm shown in Table 2.1 only nodes B, D, and E are unvisited and have weights 3, 5, and ∞ respectively. Therefore, the algorithm would move onto node B and mark it as visited. This cycle of weight evaluation, updating, and movement is repeated until all nodes have been visited. Once finished, the algorithm can determine the optimal (e.g., shortest distance) route between any of the nodes in the graph [31].

Table 2.1: Dijkstra's Algorithm Progression

Step	**A**	**B**	**C**	**D**	**E**
1	0	∞	∞	∞	∞
2	0	4	2	∞	∞
3	0	3	2	5	∞
4	0	3	2	4	5
5	0	3	2	4	5
6	0	3	2	4	5

This algorithm is naturally very useful for navigation, but like most deliberative

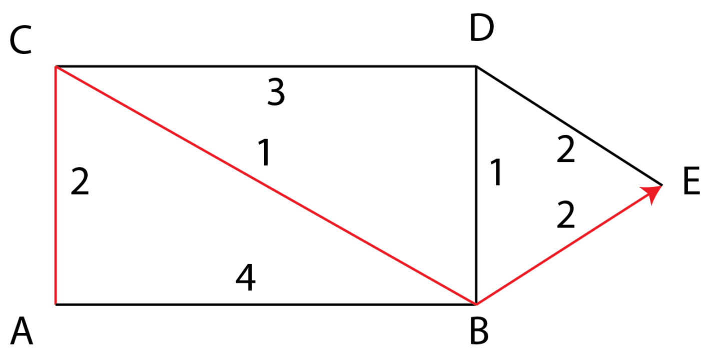


Figure 2.6: In Dijkstra's algorithm, the environment is represented as a graph of nodes connected by weighted edges. The weight can represent any cost, but is most typically associated with distance. In this particular case, the shortest path from node A to node E is highlighted in red.

approaches it is best used in predictable environments. Even so, it has been applied successfully in applications such as routing emergency vehicles and autonomous cars [32, 33], as well as airline network planning [34].

Improving upon the ideas presented by Dijkstra's algorithm, the A* algorithm was developed in 1967 by Peter Hart. The A* algorithm works much like Dijkstra's, but incorporates the estimated distance from any node to the desired finish node in the decision making process for which node to explore. We can see then, that this algorithm relies on information from the problem domain to calculate this estimate for each node in the graph. The algorithm functions much like Dijkstra's in the sense that it calculates a weight for each connected node, and moves to the nearest (if weight is determined by distance) unvisited node. It differs in that the weight is a summation of the Dijkstra weight and the previously calculated estimate. In this way, the algorithm can search for the optimal path much faster than Dijkstra's algorithm can [35]. The applications for the A* algorithm are very similar to those which utilized Dijkstra's algorithm, such as route finding on roads [36], so the two are often compared.

While these deliberative algorithms are quite good for applications such as cars on roads, where the roads are not likely to change, a more reactive approach is generally required for more agile robots who are exploring unknown or changing environments.

2.3.2 Reactive Algorithms

Reactive algorithms closely couple sensors and actuators, with minimal deliberation. For example, a very simple reactive controller on a ground robot equipped with touch sensors on either side might react to a touch on the right by turning left. This example is so simple in fact that there is no need for any decision making at all and the behavior could be achieved by simple circuitry. An early example of such a system is William Grey Walter's tortoise robot, developed in 1949, which is a simple collection of light and touch sensors coupled with clever circuitry to the robots actuators [37, 38].

Because of their simplicity, reactive controllers do not generally achieve an optimal route, and may even fail in finding a route at all. However, an element of early reactive controllers are incorporated in many modern control systems.

2.3.3 Modern UAS Control

Most modern controllers are some form of hybrid between deliberative and reactive methods. In this way, they are able to both react quickly if necessary, but can also perform some form of optimization when choosing their trajectory. For example, one group has very successfully implemented a trajectory generation technique based on minimizing

$$\int \left(\frac{s}{\|s\|}\right)^2 \tag{2.1}$$

where s is the snap, or the second derivative of the robot's acceleration [39, 40].

Given keyframes consisting of a position in space coupled with a yaw angle this method is able to generate very smooth and optimal trajectories.

Other groups have successfully applied methods utilizing Voronoi diagrams [27,41] receding horizons in relatively unrestricted environments [42], high order parametric curves [43], and 3D interpolation [44].

What most of these methods have in common is that they require a fair amount of computation. For example, minimizing the integral of the square of the norm of the snap as described in the first example can require multiple iterations, and in addition requires keyframes in order to generate an appropriate trajectory. Given a powerful enough processor, these calculations can be made in time to react appropriately in the environment. For example, they demonstrate their quadcopter generating a trajectory and flying through a hoop which is tossed into the air.

However, many platforms are not equipped with a very powerful processor and solving complex algorithms cannot practically be performed by an offboard computer. Due to the shortcomings of reactive and deliberative methods, this thesis focuses on a navigation algorithm which is computationally inexpensive and can react quickly to the environment with the intent that it could be deployed onboard any platform with the ability to sense obstacles and its own location in the environment.

2.3.4 Summary of UAS Navigation

When considering the options for a navigation system for an agile flying platform, it is clear that a hybrid solution must be used. The importance of reacting quickly to a dynamic environment in which new data is constantly acquired must be matched with finding a collision free path like a deliberative method would yield. While the methods presented here have been used with some success, their heavy computing costs and in some cases reliance upon specific sensors make them less than ideal for applications on small systems which have payload limitations and limited onboard processing power.

2.4 Summary

This chapter presented a background on unmanned autonomous systems including their importance and applications, discussed the contrast between fixed wing and multirotor platforms, and presented examples of sensing and navigation methods used in UAS systems today. The following chapter presents the model for a quadcopter, as well as the characterization of the ARDrone 2.0 used during experimentation.

Chapter 3

System Model

This section presents the set of differential equations represent the quadcopter system dynamics. Developing a mathematical model of a system is a fundamental step in any controller design and develops a deeper understanding of the system in question. Once the mathematical system and initial controller design are complete, the combined system can be simulated and tested in an experimental setting. A brief background on Newtonian reference frames is provided, as well as the method used to transform between various reference frames and describe the motion of a rigid body. Finally, the equations of motion are derived using the Newton-Euler method.

3.1 Reference Frames

To begin, it is important to introduce a set of reference frames which allow us to represent the position and orientation of a rigid body in space. These reference frames are defined by a linearly independent set of vectors which span the dimensions of the frame. The position and orientation of the rigid body at any instant in time is represented by a particle at its center of mass and is described relative to a Cartesian reference frame in Euclidean space, $E \in \mathbb{R}^3$, which is fixed on earth at a known location as shown in Fig. 3.1. It is assumed that this reference frame is non-accelerating, and is therefore inertial. Additionally, the curvature of the earth is considered negligible for the scope of this work. The axes of $E \in \mathbb{R}^3$ are described by the set of orthogonal unit vectors $(\hat{e}_x, \hat{e}_y, \hat{e}_z) \in \mathbb{R}^3$, where $\hat{e}_x$ points north, $\hat{e}_y$ points east, and $\hat{e}_z$ points toward the center of the earth.

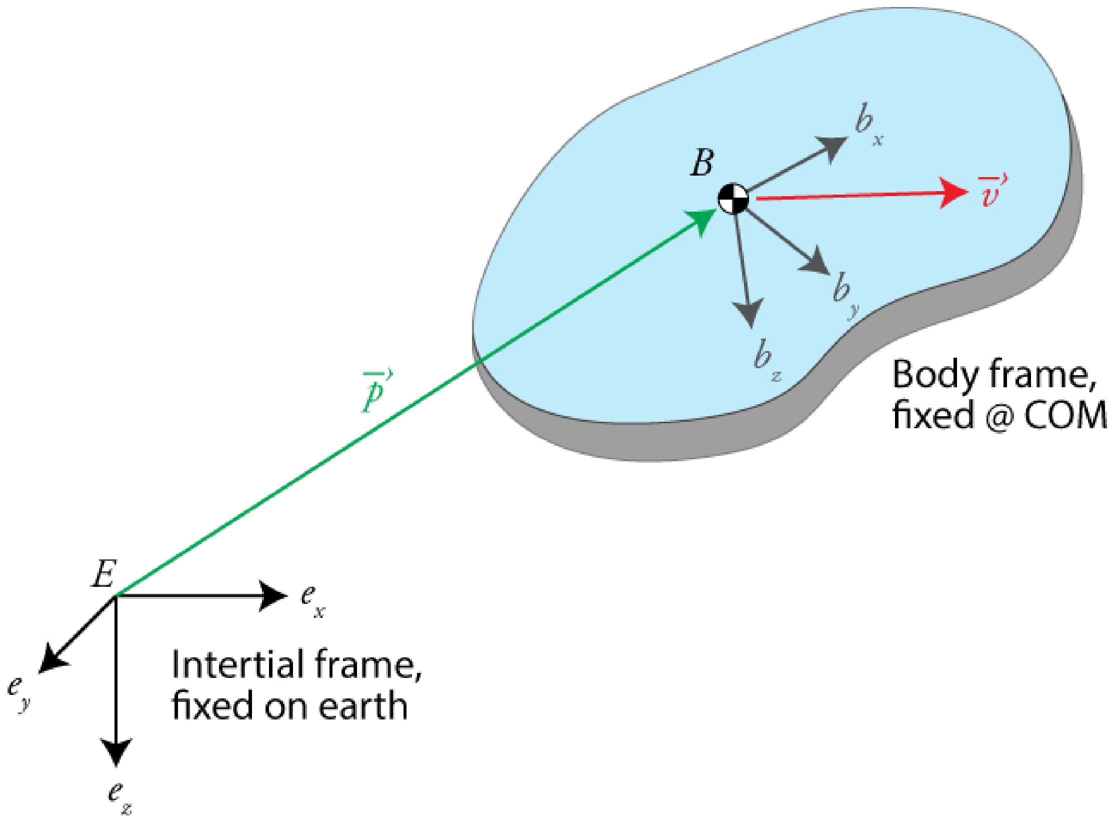


Figure 3.1: An inertial reference frame, E, is fixed on earth, and an accelerating body reference frame, B, is fixed at the rigid bodie's center of mass. The position and velocity of the rigid body are designated as $\vec{p}$ and $\vec{v}$ respectively. Euler rotation matrices may be used to transform between frames E and B for a given orientation.

To simplify the derivation of the equations of motion, an additional reference frame, $B \in \mathbb{R}^3$, is defined which is attached to the particle at the center of mass of the rigid body. This reference frame is referred to as the body frame and is represented by the set of orthogonal unit vectors $(\hat{b}_x, \hat{b}_y, \hat{b}_z) \in \mathbb{R}^3$ where $\hat{b}_x$ points forward, $\hat{b}_y$ points right, and $\hat{b}_z$ points downward, perpendicular to the body. To describe the orientation of the body, we name rotation about the $\hat{b}_x$ to be roll, $\vec{\phi}$, rotation about $\hat{b}_y$ to be pitch, $\vec{\theta}$, and rotation about $\hat{b}_z$ to be yaw, $\vec{\psi}$. It is important to note that the body frame is non-inertial and does experience acceleration.

3.2 Euler Transformations

In order to represent a vector in either reference frame, a transformation must be established between the two frames. Various methods exist for performing a transformation between frames, including Euler rotation matrices, quaternion transformations, and angle-axis representation. For the scope of this work, Euler rotation matrices are used, but their limitations are noted.

Three matrices fully describe the transformation between the body frame and the inertial frame: rotation about the $\hat{b}_z$ axis, rotation about the $\hat{b}_x$ axis, and rotation about the $\hat{b}_y$ axis. Rotation about the $\hat{b}_z$ axis (yaw) is a familiar example, common in two dimensional transformations, and can be described by

$$\mathbf{R}_\psi = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & sin(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

Similarly, rotation about the $\hat{b}_x$ axis (roll) is described by

$$\mathbf{R}_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix}. \tag{3.2}$$

Finally, rotation about the $\hat{b}_y$ axis (pitch) is described by

$$\mathbf{R}_\theta = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}. \tag{3.3}$$

While these three matrices fully described the transformation between the two coordinate frames, it is often more convenient to combine them into a single matrix for performing calculations. This final matrix is given by the product of (3.1), (3.2), and (3.3) which yields

$$\mathbf{R}_{\phi,\theta,\psi} = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\phi S_\theta - S_\psi C_\phi & C_\psi C_\phi S_\theta + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\phi S_\theta + C_\psi C_\phi & S_\psi C_\phi S_\theta - C_\psi S_\phi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix}. \tag{3.4}$$

where $S_x = sin(x)$ and $C_x = cos(x)$. A useful property of this rotation matrix is that $\mathbf{R}^{-1}_{\phi,\theta,\psi} = \mathbf{R}^T_{\phi,\theta,\psi}$ which can be used for transforming from the inertial frame to the body frame if needed. However, it should be noted that if $cos(\theta) = 0$, a singularity occurs in $\mathbf{R}_{\phi,\theta,\psi}$ in which case one degree of freedom is lost. To address this shortcoming, other methods such as quaternion representations are often used when describing aerial robots. However, for the scope of this work, it is assumed that the quadcopter will not see large angles and therefore will not experience gimbal lock.

3.3 Newton-Euler Equations

A classic method of deriving the equations of motion in robotics is the use of the Newton-Euler equations. Combined, these equations fully describe both the translational and rotational dynamics of a rigid body.

Consider the reference frame B which is attached to the particle at the robot's center of mass as discussed in Section 3.1. The velocity of B in E is defined as

$$\vec{v} = \frac{d\vec{p}}{dt}, \tag{3.5}$$

where $\vec{p}$ is the position vector from the origin of E to the origin of B at the robot's center of mass. The translational momentum of B is given by

$$\vec{L} = m\vec{v}, \tag{3.6}$$

where m is the mass of the robot. Newton's second law states that the sum of the external forces acting upon an object equals the time rate of change of its translational momentum. Therefore, the translational dynamics of the robot can be described by

$$\begin{aligned} \sum \vec{F} &= \frac{d}{dt}(\vec{L}), \\ &= \frac{d}{dt}(m\vec{v}). \end{aligned} \tag{3.7}$$

For the scope of this thesis, it is assumed that mass is time-invariant. Taking the derivative in the earth frame yields

$$\begin{aligned} \sum \vec{F}_E &= m\frac{d\vec{v}}{dt_E}, \\ &= m\vec{a}_E, \end{aligned} \tag{3.8}$$

where $\vec{a}_E$ is linear acceleration in the earth frame. The derivative can also be taken in B in order to represent the dynamics in the body frame which yields

$$\begin{aligned} \sum \vec{F}_B &= m\left(\frac{d\vec{v}}{dt_B} + \vec{\omega}_B \times \vec{v}_B\right), \\ &= m(\vec{a}_B + \vec{\omega}_B \times \vec{v}_B). \end{aligned} \tag{3.9}$$

Because B is located at the center of mass of the body, $\vec{\omega}_B$ is zero and (3.9) simplifies to be

$$\sum \vec{F}_B = m\vec{a}_B. \tag{3.10}$$

While (3.10) accurately describes the translational dynamics of the robot, the angular dynamics must still be addressed. The angular momentum of B is defined as

$$\vec{H} = I_{cm}\vec{\omega}, \tag{3.11}$$

where I_{cm} is the moment of inertia about the robot's center of mass. Euler's second law states that the sum of the torques acting upon an object equals the time rate of change its angular momentum. Therefore, we describe the rotational dynamics using

$$\begin{aligned} \sum \vec{\tau} &= \frac{d}{dt}(\vec{H}), \\ &= \frac{d}{dt}(I_{cm}\vec{\omega}). \end{aligned} \tag{3.12}$$

Similar to mass, we assume that I_{cm} is time-invariant. Therefore, the time derivative of $\vec{H}$, taking into account a rotating frame, is found as

$$\begin{aligned} \sum \vec{\tau}_B &= I_{cm}\frac{d\vec{\omega}}{dt}_B + \vec{\omega}_B \times I_{cm}\vec{\omega}_B, \\ &= I_{cm}\vec{\alpha}_B + \vec{\omega}_B \times I_{cm}\vec{\omega}_B, \end{aligned} \tag{3.13}$$

where $\vec{\alpha}$ is angular acceleration.

It is common to combine (3.10) and (3.13) into matrix form for a more compact representation, given by

$$\begin{bmatrix} \sum \vec{F}_B \\ \sum \vec{\tau}_B \end{bmatrix} = \begin{bmatrix} mI_3 & 0 \\ 0 & I_{cm} \end{bmatrix} \begin{bmatrix} \vec{a}_B \\ \vec{\alpha}_B \end{bmatrix} + \begin{bmatrix} 0 \\ \vec{\omega}_B \times I_{cm}\vec{\omega}_B \end{bmatrix}, \tag{3.14}$$

where I_3 is a $3x3$ identity matrix. This is the classic form of the Newton-Euler equations for a system with B oriented at the system's center of mass, and will be the basis for determining the equations of motion specific to the quadcopter platform.

3.4 Quadcopter Dynamics

The quadcopter platform shown in Fig 3.2 is assumed to be symmetric about the $\hat{b}_x$ and $\hat{b}_y$ axis. Thus the inertia matrix in the body frame is given by

$$I_{cm} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}, \tag{3.15}$$

where I_{xx} and I_{yy} are equal due to symmetry.

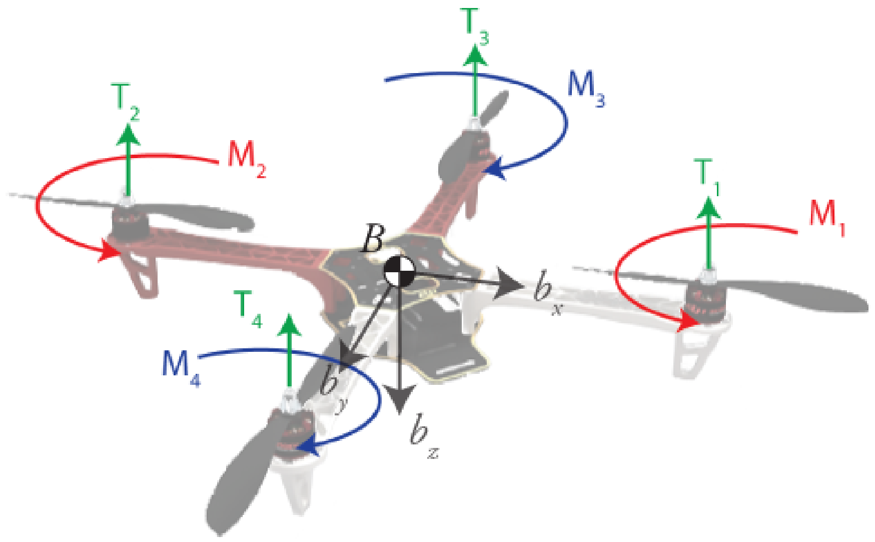


Figure 3.2: A quadcopter is symmetric about both the $\hat{b}_x$ and $\hat{b}_y$ axis. Each motor produces a force, $\vec{T_i}$ and moment, $\vec{M_i}$, which are functions of the motor's angular velocity, Ω_i.

Each of the four motors on the quadcopter has an associated force associated with it, given by

$$\vec{T_i} = -k_T \Omega_i^2 \hat{b_z}, \tag{3.16}$$

where $\vec{T_i}$ is the force (or thrust) provided by the i^{th} motor, Ω_i is the angular velocity of the motor, and k_T is a constant which is a function of the specific motor and propeller used. For the scope of this thesis, it is assumed that the density of air remains constant and that roll and pitch are small angles. Using this, sum of the forces is found to be

$$\sum \vec{F} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & mg - k_T \sum_{i=1}^{4} \Omega_i^2 \end{bmatrix} \begin{bmatrix} \hat{b_x} \\ \hat{b_y} \\ \hat{b_z} \end{bmatrix}. \tag{3.17}$$

Similar to thrust, each motor also has an associated moment, given by

$$\vec{M_i} = -k_M \Omega_i^2 \hat{b_z}, \tag{3.18}$$

where $\vec{M_i}$ is the moment generated by the i^{th} motor, and k_M is a constant which is again a function of the specific motor and propeller used. In addition to the moments generated by the motors themselves, there are torque contributions from the moment arms produced by the motor's forces. The sum of the torques is found to be

$$\sum \vec{\tau} = \begin{bmatrix} k_T(\Omega_3^2 - \Omega_4^2)l & 0 & 0 \\ 0 & k_T(\Omega_1^2 - \Omega_2^2)l & 0 \\ 0 & 0 & -k_M(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \end{bmatrix} \begin{bmatrix} \hat{b_x} \\ \hat{b_y} \\ \hat{b_z} \end{bmatrix}, \tag{3.19}$$

where l is the length of the quadcopter's arms.

From (3.17) and (3.19), we can see that linear translational dynamics are closely coupled with the rotational dynamics. This is expected because a quadcopter is an underactuated system, having only four actuators and six degrees of freedom.

Using the result of (3.17) and (3.19) in the Newton-Euler equations, we find that the dynamics of the system in the body frame, B, and assuming small roll and pitch angles are described by

$$\begin{bmatrix} \vec{a}_x \\ \vec{a}_y \\ \vec{a}_z \\ \vec{\alpha}_x \\ \vec{\alpha}_y \\ \vec{\alpha}_z \end{bmatrix} = \begin{bmatrix} \frac{1}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \left(\begin{bmatrix} 0 \\ 0 \\ mg - k_T \sum_{i=1}^{4} \Omega_i^2 \\ k_T(\Omega_3^2 - \Omega_4^2)l \\ k_T(\Omega_1^2 - \Omega_2^2)l \\ -k_M(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dot{\theta}\dot{\psi}(I_{zz} - I_{yy}) \\ \dot{\phi}\dot{\psi}(I_{xx} - I_{zz}) \\ \dot{\theta}\dot{\phi}(I_{yy} - I_{xx}) \end{bmatrix} \right)$$
$$= \begin{bmatrix} 0 \\ 0 \\ g - \frac{k_T}{m} \sum_{i=1}^{4} \Omega_i^2 \\ \frac{1}{I_{xx}} \left(k_T(\Omega_3^2 - \Omega_4^2)l - \dot{\theta}\dot{\psi}(I_{zz} - I_{yy}) \right) \\ \frac{1}{I_{yy}} \left(k_T(\Omega_1^2 - \Omega_2^2)l - \dot{\phi}\dot{\psi}(I_{xx} - I_{zz}) \right) \\ \frac{1}{I_{zz}} \left(-k_M(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) - \dot{\theta}\dot{\phi}(I_{yy} - I_{xx}) \right) \end{bmatrix}. \tag{3.20}$$

These equations fully describe the motion of the quadcopter platform in the body coordinates, and can be used for simulating the dynamics of the platform to evaluate controller performance.

3.5 Summary

This chapter presented a background on the concepts of reference frames used, a review of the methods of transforming between the two reference frames, as well as a background on the derivation of the Newton-Euler equations. Finally, the differential equations of motion for the quadcopter platform were derived, which can be used in for simulating the response of the system to the controller designed in Chapter 4.

Chapter 4

Design of an ePFC

This chapter presents the development of an extended potential field controller which builds upon the concepts used by traditional potential field controllers. Therefore, a background on traditional potential field controllers is presented in Section 4.1, after which the extended portion of the controller is developed in Section 4.2. Finally, the stability of the controller is analyzed in Section 4.3.

4.1 Traditional Potential Field Controller

Because of their simplicity and elegance, potential field methods are often used for navigation of ground robots [45–47]. Potential fields are aptly named, because they use attractive and repulsive potential equations to draw the drone toward a goal (attractive potential) or push it away from an obstacle (repulsive potential). For example, imagine a stretched spring which connects a drone and a target. Naturally, the spring draws the drone to the target location.

Conveniently, potential fields for both attractive and repulsive forces can be summed together, to produce a field such as the one shown in Fig. 4.1. This figure illustrates how a robot can navigate toward a target location while simultaneously avoiding obstacles in its path.

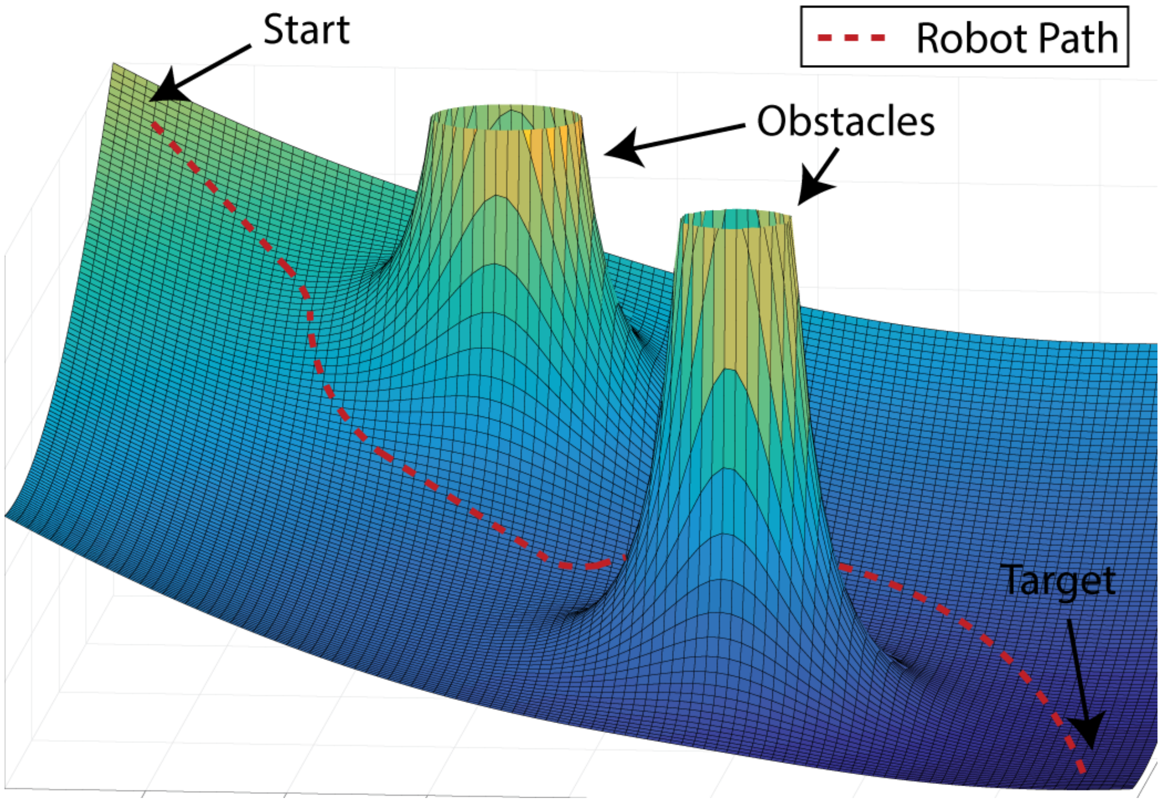


Figure 4.1: An example of a traditional potential field which can be used for navigating toward a target while avoiding multiple obstacles.

Let us denote $\vec{p}_d = [x_d, y_d, z_d]^T$ and $\vec{p}_t = [x_t, y_t, z_t]^T$ as the position vector of drone and target, respectively. The relative distance vector between the drone and the target is then

$$\begin{aligned}\vec{p}_{dt} &= [x_{dt}, y_{dt}, z_{dt}]^T, \\ &= [x_d, y_d, z_d]^T - [x_t, y_t, z_t]^T.\end{aligned} \tag{4.1}$$

Traditionally, potential forces work in the x, y, and z spatial dimensions, and are defined by a quadratic function given by

$$U_{att^1}(\vec{p}_d, \vec{p}_t) = \frac{1}{2}\lambda_1 \|\vec{p}_{dt}\|^2, \tag{4.2}$$

where λ_1 is positive scale factor, and $\|\vec{p}_{dt}\|$ is the magnitude of the relative distance between the drone and the target, which is given by

$$\|\vec{p}_{dt}\| = \sqrt{(x_{dt})^2 + (y_{dt})^2 + (z_{dt})^2}. \tag{4.3}$$

As shown in Fig. 4.1, the target location is always a minimum, or basin, of the overall potential field. Therefore, in order to achieve the target location, the UAS should always move “downhill.” The direction and magnitude of the desired movement can be computed by finding the negative gradient of the potential field, given by

$$\begin{aligned}
\vec{v}_d^{att^1}(\vec{p}_d, \vec{p}_t) &= -\nabla U_{att^1}(\vec{p}_d, \vec{p}_t), \\
&= -\frac{\partial U_{att^1}}{\partial x} - \frac{\partial U_{att^1}}{\partial y} - \frac{\partial U_{att^1}}{\partial z}, \\
&= -\nabla\Big(\frac{1}{2}\lambda_1 \|\vec{p}_{dt}\|^2\Big), \\
&= -\frac{1}{2}\lambda_1 \nabla \|\vec{p}_{dt}\|^2, \\
&= -\frac{1}{2}\lambda_1 \nabla (x_{dt}^2 + y_{dt}^2 + z_{dt}^2), \\
&= -\lambda_1 (x_{dt} + y_{dt} + z_{dt}), \\
&= -\lambda_1 (\vec{p}_d - \vec{p}_t),
\end{aligned} \tag{4.4}$$

where $\vec{v}_d^{att^1}$ is the desired velocity due to the attractive position potential.

This is the classic form of a simple attractive potential field controller. However, this does not yet take into account obstacles or other sources of repulsive potential. The repulsive potential is proportional to the inverse square of the distance between the drone and the obstacle and is given by

$$U_{rep^1}(\vec{p}_d, \vec{p}_o) = \frac{1}{2}\eta_1 \frac{1}{\|\vec{p}_{do}\|^2}, \tag{4.5}$$

where η_1 is positive scale factor, and $\|\vec{p}_{do}\|$ is the magnitude of the relative distance between the drone and the obstacle.

To find the desired velocity, we again take the gradient of the potential field which yields

$$\begin{aligned}
\vec{v}_d^{rep^1}(\vec{p}_d, \vec{p}_o) &= -\nabla U_{rep^1}(\vec{p}_d, \vec{p}_o), \\
&= -\frac{\partial U_{rep^1}}{\partial x} - \frac{\partial U_{rep^1}}{\partial y} - \frac{\partial U_{rep^1}}{\partial z}, \\
&= -\nabla \Big(\frac{1}{2}\eta_1 \frac{1}{\|\vec{p}_{do}\|^2}\Big), \\
&= -\frac{1}{2}\eta_1 \nabla \frac{1}{\|\vec{p}_{do}\|^2}, \\
&= -\frac{1}{2}\eta_1 \nabla \frac{1}{x_{do}^2 + y_{do}^2 + z_{do}^2}, \\
&= \eta_1 \frac{\vec{p}_d - \vec{p}_o}{(x_{do} + y_{do} + z_{do})^2}, \\
&= \eta_1 \frac{\vec{p}_d - \vec{p}_o}{\|\vec{p}_{do}\|^3},
\end{aligned} \tag{4.6}$$

where $\vec{v}_d^{rep^1}$ is the desired velocity due to the repulsive position potential.

It is important to note that the repulsive potential is designed to have no effect when the drone is more than a set distance away from the obstacle, P^*, as shown in Fig. 4.2.

Therefore, the velocity due to repulsive potentials becomes

$$\vec{v}_d^{rep^1}(\vec{p}_d, \vec{p}_o) = \begin{cases} \eta_1 \frac{\vec{p}_d - \vec{p}_o}{\|\vec{p}_{do}\|^3}, & \|\vec{p}_{do}\| \leq P^* \\ 0, & \|\vec{p}_{do}\| > P^*. \end{cases} \tag{4.7}$$

A complete traditional potential field controller is the sum of (4.4) and (4.7) which

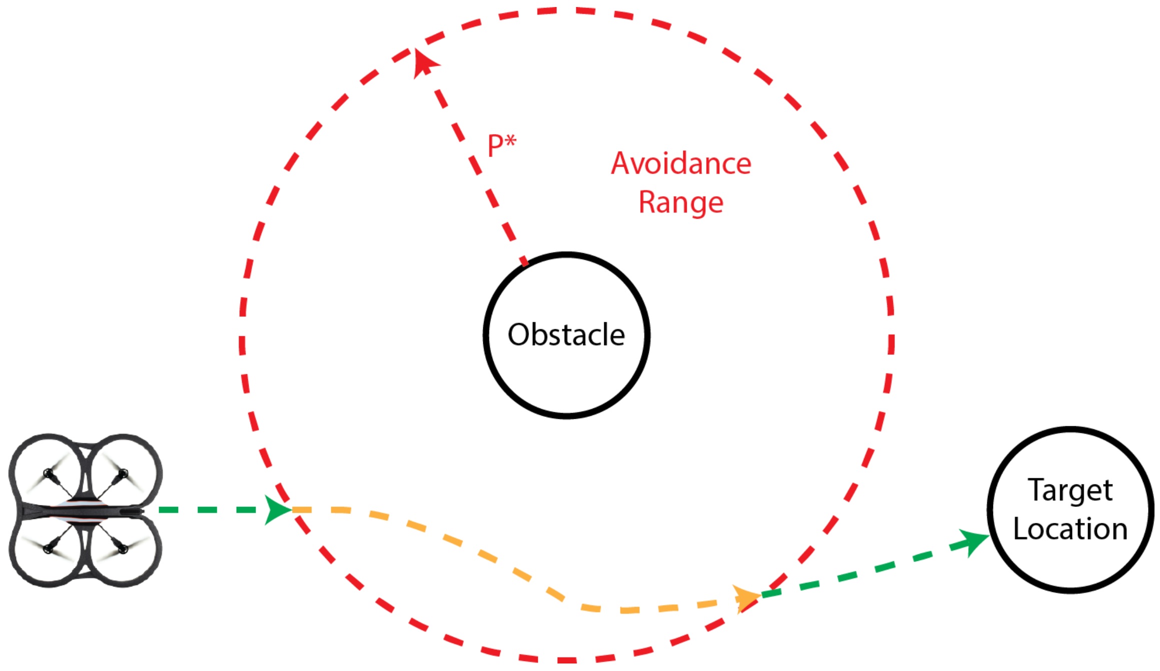


Figure 4.2: The repulsive force due to an obstacle should be zero when the drone is sufficiently far from it. In this case, the limit is P^*. While outside of this limit, the drone is unaffected by the obstacle's repulsive potential. However, as soon as the relative distance becomes less than P^*, the drone takes avoidance action.

yields

$$\vec{v}_d^{PFC}(\vec{p}_d, \vec{p}_t, \vec{p}_o) = \begin{cases} -\lambda_1(\vec{p}_t - \vec{p}_d) + \sum_{i=0}^{n} \eta_1 \frac{\vec{p}_d - \vec{p}_o^{\,i}}{\|\vec{p}_{do}^{\,i}\|^3}, & \|\vec{p}_{do}^{\,i}\| \leq P^* \\ -\lambda_1(\vec{p}_t - \vec{p}_d), & \|\vec{p}_{do}^{\,i}\| > P^*. \end{cases} \tag{4.8}$$

where n is the number of obstacles present in the environment.

This controller enables a ground robot to track stationary or dynamic targets, while avoiding any obstacles in its path. However, when applied to an agile, aerial system such as a quadcopter, the controller's performance is quite poor as shown in simulations presented in Chapter 5.

4.2 Extended Potential Field Controller

Because the potential field methods presented above are developed for ground robots, they do not address many of the factors that must be accounted for when designing a controller for aerial systems. For example, drones move very quickly and are inherently unstable which means they cannot simply move to a particular location and stop moving. They are consistently making fine adjustments to their position and velocity.

In order to account for factors unique to aerial platforms, this thesis presents an extended potential field controller (ePFC) which utilizes the same concepts found in a traditional PFC, but applied to relative velocities rather than positions. If we consider that we are tracking a dynamic target, then the desired velocity will be that of the target. In this case, the attractive potential will be defined as the quadratic function given by

$$U_{att^2}(\vec{v}_d, \vec{v}_t) = \frac{1}{2}\lambda_2 \|\vec{v}_{dt}\|^2, \tag{4.9}$$

where λ_2 is positive scale factor, and $\|v_{dt}\|$ is the magnitude of the relative velocity between the drone velocity, v_d, and the target velocity, v_t, which is given by

$$\|\vec{v}_{dt}\| = \sqrt{(\dot{x}_{dt})^2 + (\dot{y}_{dt})^2 + (\dot{z}_{dt})^2}. \tag{4.10}$$

As in the traditional potential field controller, we wish to minimize the relative velocity potential thus resulting in a matched velocity between the drone and the target. Similar to the traditional controller, we find the desired velocity of the drone by calculating the negative gradient, which we find to be

$$\begin{aligned}
\vec{v}_d^{att^2}(\vec{v}_d,\vec{v}_t) &= -\nabla U_{att^2}(\vec{v}_d,\vec{v}_t), \\
&= -\frac{\partial U_{att^2}}{\partial \dot{x}} - \frac{\partial U_{att^2}}{\partial \dot{y}} - \frac{\partial U_{att^2}}{\partial \dot{z}}, \\
&= -\nabla\Big(\frac{1}{2}\lambda_2\|\vec{v}_{dt}\|^2)\Big), \\
&= -\frac{1}{2}\lambda_2\nabla\|\vec{v}_{dt}\|^2, \\
&= -\frac{1}{2}\lambda_2\nabla(\dot{x}_{dt}^2+\dot{y}_{dt}^2+\dot{z}_{dt}^2), \\
&= -\lambda_2(\dot{x}_{dt}+\dot{y}_{dt}+\dot{z}_{dt}), \\
&= -\lambda_2(\vec{v}_d-\vec{v}_t).
\end{aligned} \tag{4.11}$$

If we consider that the drone and an obstacle should not maintain the same velocity, then we can design the repulsive velocity potential between the drone and an obstacle to be an inverse quadratic as in (4.5), given by

$$U_{rep^2}(\vec{v}_d,\vec{v}_o) = \frac{1}{2}\eta_2\frac{1}{\|\vec{v}_{do}\|^2}, \tag{4.12}$$

where η_2 is positive scale factor, and $\|\vec{v}_{do}\|^2$ is the magnitude of the relative velocity between the drone velocity, $\vec{v}_d$, and the obstacle velocity, $\vec{v}_o$. The corresponding velocity for this potential function is found by

$$\begin{aligned}
\vec{v}_d^{rep^2}(\vec{v}_d, \vec{v}_o) &= \nabla U_{rep^2}(\vec{v}_d, \vec{v}_o), \\
&= \frac{\partial U_{rep^2}}{\partial \dot{x}} + \frac{\partial U_{rep^2}}{\partial \dot{y}} + \frac{\partial U_{rep^2}}{\partial \dot{z}}, \\
&= -\nabla\Big(\frac{1}{2}\eta_2 \frac{1}{\|\vec{v}_{do}\|^2}\Big), \\
&= -\frac{1}{2}\eta_2 \nabla \frac{1}{\|\vec{v}_{do}\|^2}, \\
&= -\frac{1}{2}\eta_2 \nabla \frac{1}{\dot{x}_{do}^2 + \dot{y}_{do}^2 + \dot{z}_{do}^2}, \\
&= \eta_2 \frac{\vec{v}_d - \vec{v}_o}{(\dot{x}_{do}^2 + \dot{y}_{do}^2 + \dot{z}_{do}^2)^2}, \\
&= \eta_2 \frac{\vec{v}_d - \vec{v}_o}{\|\vec{v}_{do}\|^3}.
\end{aligned} \tag{4.13}$$

It should be noted that if the obstacle is stationary, then its velocity will be zero. In this special case, the repulsive field in (4.12) is designed to have no effect on the drone's motion. Therefore, the velocity found in (4.13) becomes

$$\vec{v}_d^{rep^2}(\vec{v}_d, \vec{v}_o) = \begin{cases} \eta_2 \frac{\vec{v}_d - \vec{v}_o}{\|\vec{v}_{do}\|^3}, & \|\vec{v}_o\| \neq 0 \\ 0, & \|\vec{v}_o\| = 0. \end{cases} \tag{4.14}$$

Finally, the relative distance between the drone and obstacle, $\vec{p}_{do}$, is revisited. In the traditional potential field controller presented in Section 4.1, $\vec{p}_{do}$ was used as the basis for a repulsive potential. However, no thought is given to the time rate of change of the magnitude of $\vec{p}_{do}$. If we consider a situation in which the drone is moving away from the obstacle, then $\|\dot{p}_{do}\| \geq 0$ in which case no avoidance action needs to be taken, even if the drone is within the avoidance range. As illustrated in Fig. 4.3, the controller is able to ignore the effect of the obstacle sooner, and therefore can take a more direct route, thus saving time.

Furthermore, we consider that the drone should take evasive action if $\|\dot{p}_{do}\| < 0$. Therefore, a final control effort is designed as

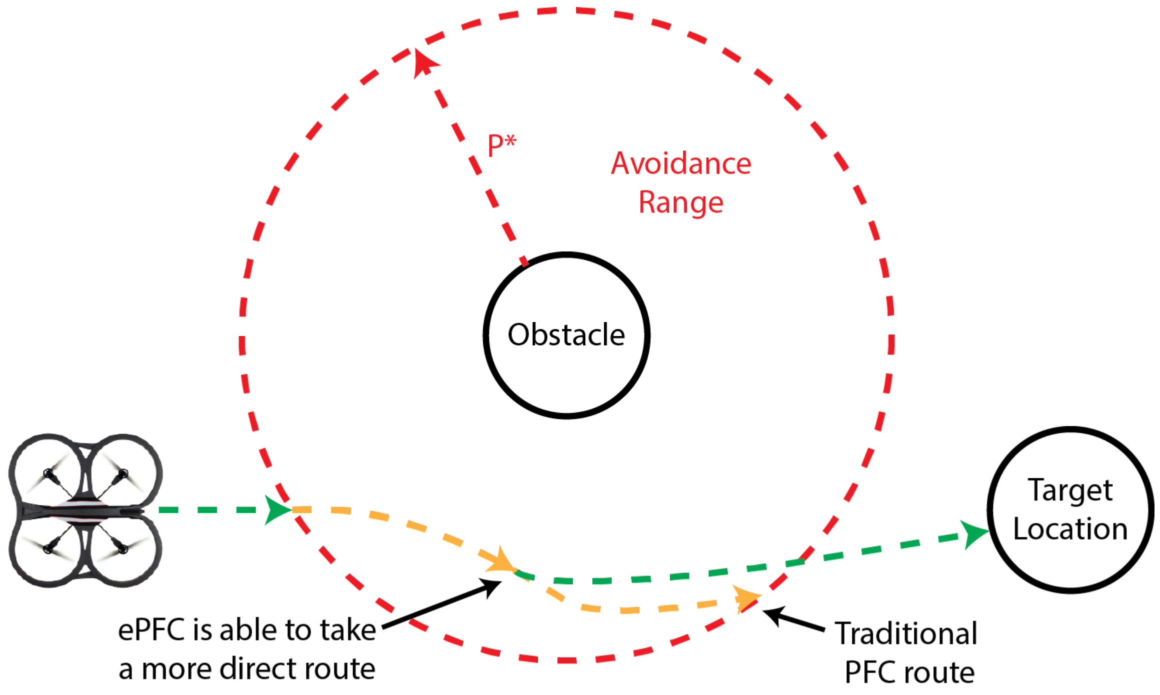


Figure 4.3: By taking into account the time rate of change of $\|\vec{p}_{do}\|$, the controller is able to ignore repulsive effects from the obstacle if the drone is moving away from the obstacle ($\|\dot{p}_{do}\| \geq 0$). This results in a more direct route to the target, thus saving time.

$$\vec{v}_d^{rep^3}(\vec{p}_d, \vec{p}_o) = \begin{cases} -\eta_3 \|\dot{p}_{do}\| \frac{\vec{p}_{do}}{\|\vec{p}_{do}\|}, & \|\dot{p}_{do}\| < 0 \\ 0, & \|\dot{p}_{do}\| \geq 0, \end{cases} \tag{4.15}$$

where η_3 is positive scale factor.

Summing the velocities in (4.11), (4.14), and (4.15) with the traditional controller (4.8) yields the full form of the extended potential field controller (ePFC), which is

$$\vec{v}_d^{ePFC} = \vec{v}_d^{PFC} - \lambda_2(\vec{v}_d - \vec{v}_t) + \sum_{i=0}^{n} \eta_2 \frac{\vec{v}_d - \vec{v}_o^i}{\|\vec{v}_{do}^i\|^3} - \sum_{i=0}^{n} \eta_3 \|\dot{p}_{do}^i\| \frac{\vec{p}_{do}^i}{\|\vec{p}_{do}^i\|}, \tag{4.16}$$

where n is the number of obstacles present, $\|v_o^i\| \neq 0$, $\|\dot{p}_{do}^i\| < 0$, and the same conditions discussed in Section 4.1 apply to v_d^{PFC}.

Finally, the velocity found in (4.16) must be transformed into the body coordinate system of the drone and is found to be

$$\vec{v}_{d,body}^{ePFC} = \vec{v}_d^{ePFC} * \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & sin(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.17}$$

where ψ is the yaw angle of the drone around the body z axis.

This controller will seek out a moving target, and will also avoid obstacles that are in close proximity.

4.3 Stability Analysis

To analyze the convergence of the proposed velocity controller (4.16) for the drone, we use the Lyapunov theory. We can choose the Lyapunov function as follows:

$$L = U_{att} = \frac{1}{2}\lambda_1 \|p_{dt}\|^2 + \frac{1}{2}\lambda_2 \|v_{dt}\|^2. \tag{4.18}$$

This function represents the attractive potentials of the controller. The repulsive potentials are designed to be unstable and are not considered for stability analysis. We see that (4.18) is positive definite, and its lie derivative is given by

$$\begin{aligned} L^* &= \frac{\partial L}{\partial p_{dt}} v_{dt} + \frac{\partial L}{\partial v_{dt}} a_{dt}, \\ &= \lambda_1 \|p_{dt}\| v_{dt} + \lambda_2 \|v_{dt}\| a_{dt}, \end{aligned} \tag{4.19}$$

where a_{dt} is the relative acceleration between the drone acceleration and the target acceleration.

Note that the relative velocity between the drone and the target is designed following the direction of the negative gradient of $U_{att}(p_{dt})$ with respect to p_{dt} as in (4.4). From (4.11), we can obtain

$$\begin{aligned} a_{dt} = \dot{v}_{dt} &= \frac{d}{dt}\left(-\frac{1}{2}\lambda_2 \nabla(\dot{x}_{dt}^2 + \dot{y}_{dt}^2 + \dot{z}_{dt}^2)\right), \\ &= \frac{d}{dt}\left(-\lambda_2 \|v_{dt}\|\right), \\ &= -\lambda_2 \|\frac{v_{dt}(t) - v_{dt}(t-1)}{\Delta_t}\|, \end{aligned} \tag{4.20}$$

where Δ_t is a time step. Hence, substituting v_{dt} given by (4.4) and a_{dt} given by (4.20) into (4.19), we obtain

$$L^* = -\left[\lambda_1^2 \|p_{dt}\|^2 + \lambda_2^2 \|v_{dt}\| \|\frac{v_{dt}(t) - v_{dt}(t-1)}{\Delta_t}\|\right], \tag{4.21}$$

We can easily see that $L^* < 0$ since $\|p_{dt}\|$, $\|v_{dt}\|$, and $\|\frac{v_{dt}(t)-v_{dt}(t-1)}{\Delta_t}\|$ are positive. This means that the proposed controller is stable, and the drone is able to track a moving target.

4.4 Summary

This section presented a background on traditional potential field controllers, and discussed the design and development of a new extended potential field controller for use on aerial robots. Additionally, it was shown using Lyapunov stability criteria that this controller will always converge on the target location. The following section presents the methodology and results of simulating the ARDrone 2.0 using the proposed controller.

Chapter 5

Simulation

5.1 MATLAB Environment

In order to validate the developed controller, the system was simulated using a Matlab Simulink model. The state space representation of the ARDrone's platform dynamics are take from the ARDrone Simulink Development Kit [48]. The complete Simulink model shown in Fig. 5.1 demonstrates how the ePFC controller uses feedback information from the ARDrone simulation and position estimator blocks. The output of the ARDrone simulation block is simply the velocity of the drone, and the position estimator uses an integrator with zero initial conditions to calculate position.

The desired path that the drone is to take is outlined in Table 5.1. A virtual obstacle is placed at $(1, 1)$ which places it immediately in the path of the drone between waypoints 2 and 3. The drone is allowed two seconds at each waypoint in an attempt to let it settle before moving on to the next waypoint.

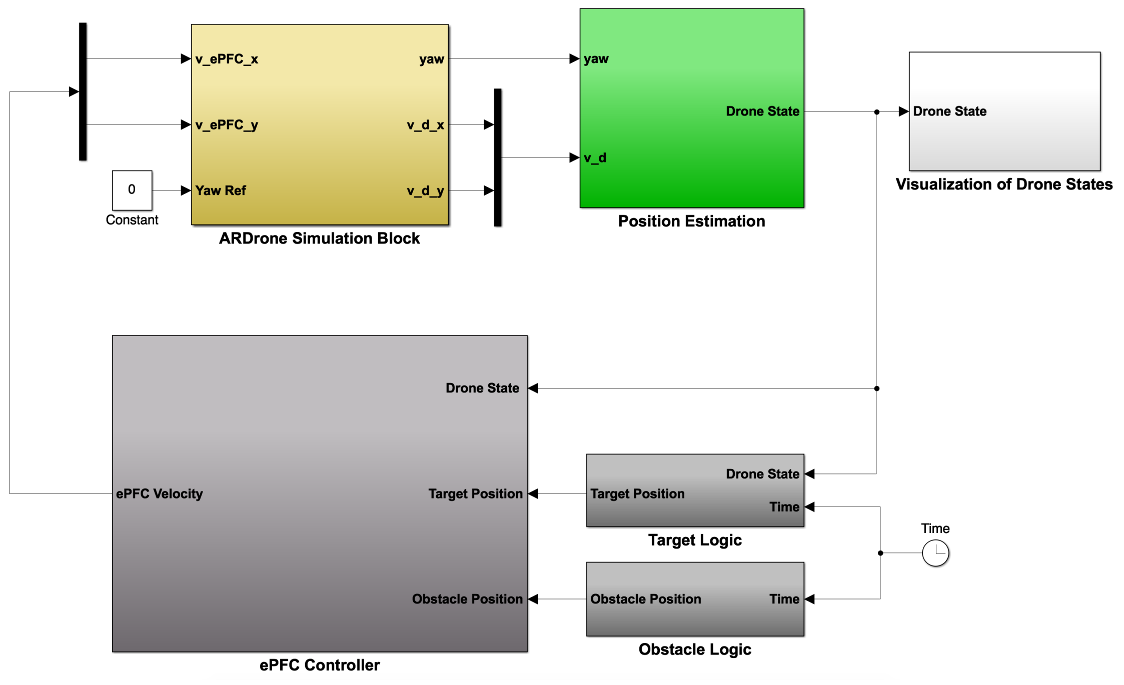


Figure 5.1: The Simulink model used includes a state space representation from the ARDrone Simulink Development Kit, as well as custom blocks for the ePFC controller described in this work.

Table 5.1: Simulation Waypoints

Waypoint	**X Coordinate [m]**	**Y Coordinate [m]**
1	2.5	-1
2	2.5	1
3	-2.5	1
4	-2.5	-1

5.2 Simulation results

First, a traditional potential field controller was simulated, and the resulting path is shown in Fig. 5.2. The performance of the traditional PFC was poor as expected, because aerial drones have very different dynamics than their ground counterparts. Using the traditional PFC, the drone overshoots the desired waypoint, and while it does avoid the obstacle at $(1, 1)$ it is not by much. The drone completed a full loop in approximately 35 seconds.

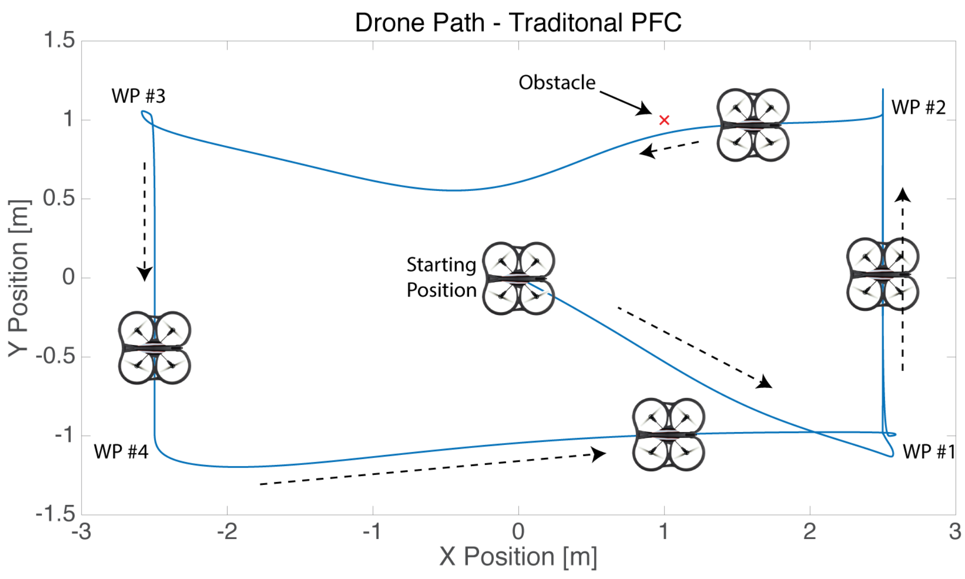


Figure 5.2: A traditional PFC is simulated on the ARDrone, with poor results. Because drones cannot stop instantaneously like ground robots, the drone often overshoots the desired waypoint. For reference, the drone takes approximately 35 seconds to complete a full loop of the course.

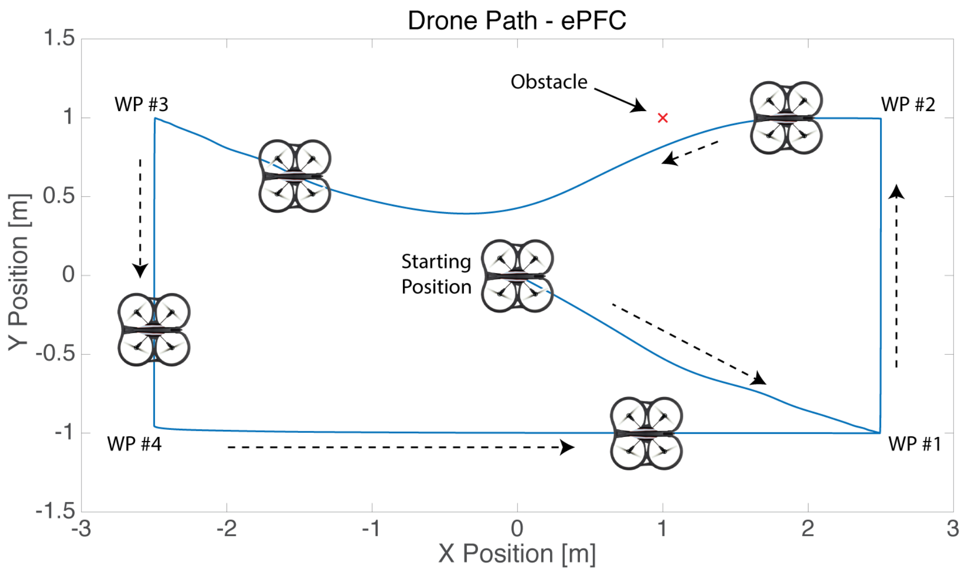


Figure 5.3: Using the extended potential field controller (ePFC), the drone is able to complete the course without overshooting the target waypoints, and avoids the obstacle by a larger margin than the traditional controller. Because the drone does not overshoot the target, it is able to complete the course in a shorter amount of time compared to the traditional controller.

Next, the ePFC is tested using the same path and obstacle position. The results shown in Fig. 5.3 demonstrate the effectiveness of the new controller. The drone does not overshoot the desired waypoints and avoids the obstacle by a larger margin, while completing the course in a shorter amount of time than the traditional controller.

Table 5.2: Simulation Controller Evaluation

Controller	**Overshoot** [%]	**Settling Time** [sec]
Traditional PFC	>19%	6
ePFC	0%	5

As outlined in Table 5.2, the ePFC controller has zero overshoot, and has a settling time of approximately five seconds. This is a large improvement over the traditional controller which overshoots by up to 19% and takes nearly six seconds to settle. It is clear that the proposed controller is more appropriate for use on an aerial drone than the traditional PFC.

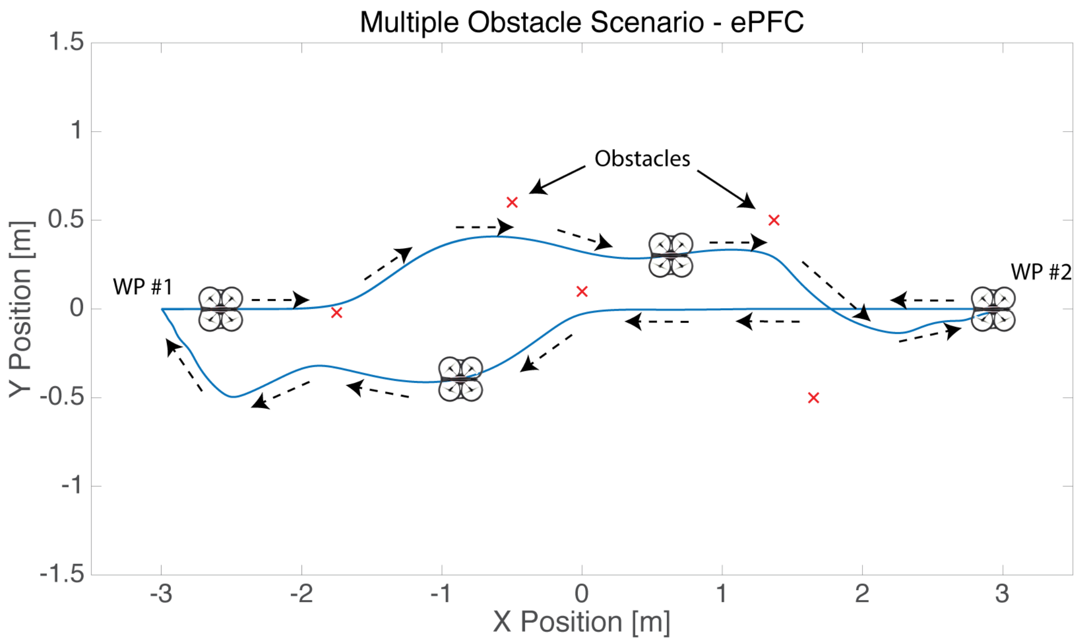


Figure 5.4: A more complex simulation was performed with multiple obstacles placed throughout the environment. The drone is able to successfully navigate between waypoints without colliding with a single obstacle, thus demonstrating its effectiveness in multi-obstacle scenarios.

In addition to the comparison between the tradition PFC and the ePFC, a more complex simulation was performed which included several obstacles placed at random throughout the environment. The results shown in Fig. 5.4 demonstrate the that the ePFC is very effective in multi-obstacle scenarios, and it successfully navigates between waypoints without colliding with a single obstacle.

5.3 Summary

This chapter presented the Matlab Simulink model that was used to simulate the drone's response to the proposed controller. In addition, both a traditional PFC and the proposed ePFC were simulated and a comparison was made. This comparison shows that the ePFC performs much better than the traditional PFC, cutting overshoot down from 19% to 0%, and reducing settling time by approximately 17%.

Chapter 6

Experimental Setup, Results, and Discussion

This chapter presents the experimental setup used to implement the proposed controller. In addition, the results from implementation are presented and the performance of the proposed controller is discussed.

6.1 Experimental Setup

The experimental platform chosen to implement the ePFC is the ARDrone 2.0 quadrotor shown in Fig. 6.1. This platform was chosen for its ease of communication - over a wifi connection - as well as the safety provided by the foam hull. Additionally, the ARDrone requires little to no setup and spare parts are readily available in case of crashes. The ARDrone 2.0 can be equipped with a 1500 mAh battery which yields flight times up to 18 min. Large batteries and long flight times are very advantageous in a testing environment because it allows for more uninterrupted tests and less

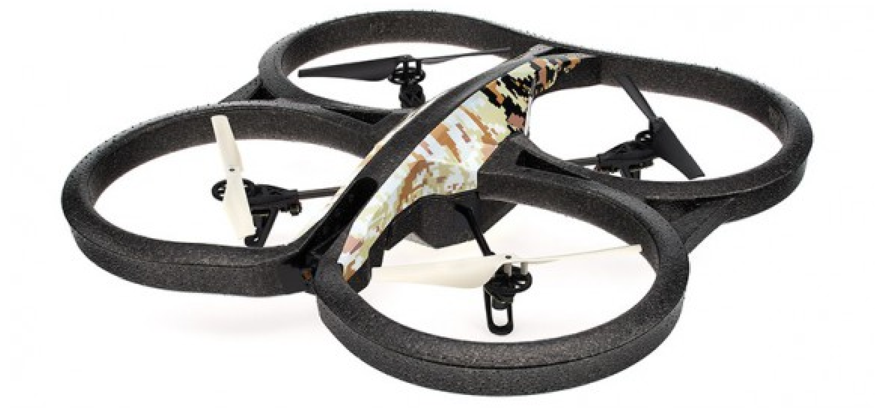


Figure 6.1: A low cost, commercially available quadrotor drone is used as the experimental platform for testing and demonstrating the effectiveness of the proposed control method.

downtime recharging batteries. The ARDrone 2.0 is also equipped with a 1 GHz 32 bit ARM Cortex A8 processor, 1 GB DDR2 RAM, and runs Linux. This means that the developed controller can be implemented onboard the drone in future work.

Sixteen Motion Analysis Kestrel cameras located throughout the testing space provide the position and orientation of the drone, target (if not virtual), and any obstacles present. The Cortex software suite provides a visual representation of the environment as shown in Fig. 6.2 as well as sending data over a network connection for use by external programs.

In order to control the drone and display its location along with the target and any obstacles present, the Matlab GUI shown in Fig. 6.3 was created. It allows the user to determine when the drone takes off, lands, or tracks the target. This GUI is critical in efficient testing of the drone. Additionally, for the safety of the drone, if the controller does not behave as expected the user can request that the drone simply hover in place to avoid fly-aways.

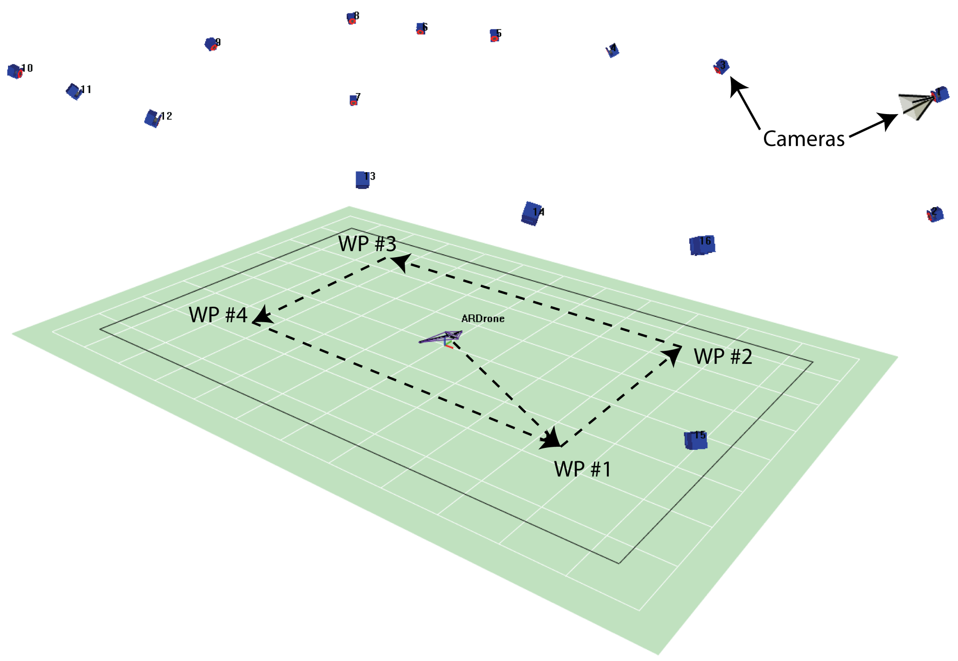


Figure 6.2: The Motion Analysis Cortex software gives the user a real-time, visual 3D representation of the environment including camera locations and any objects sensed by the system [1].

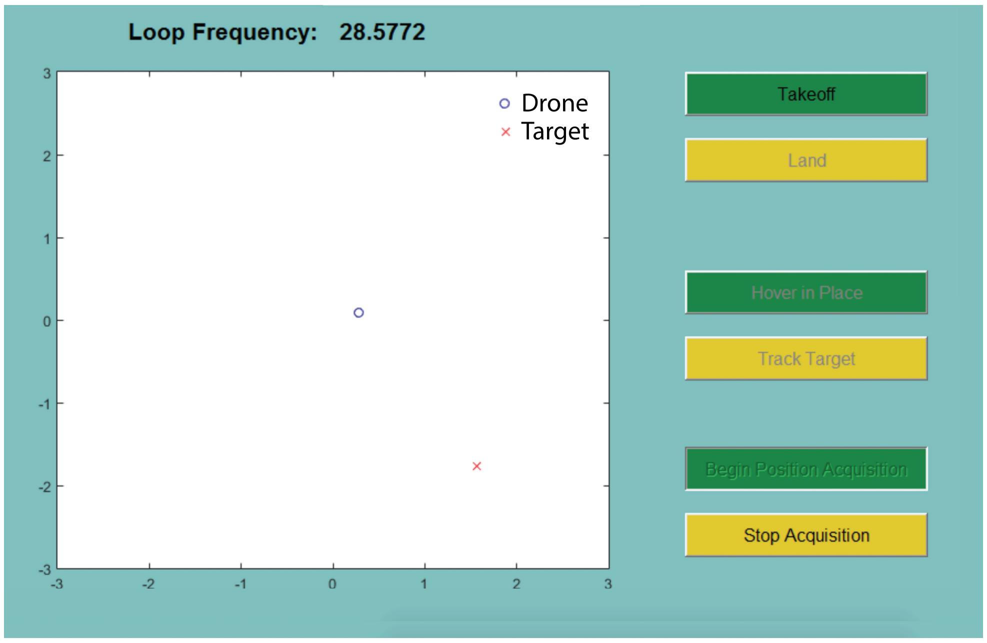


Figure 6.3: A Matlab GUI was created to show the positions of the drone, target, and any obstacles present. It also allows the user to control when the drone takes off, lands, tracks the target, or simply hovers in place.

The overview of the experimental setup shown in Fig. 6.4 demonstrates the feedback loop implemented. The Motion Analysis external tracking system is used for localization of the drone and obstacles in real time. The position information is used by the same Simulink model shown in Chapter 5 which controls the ARDrone over a wireless connection.

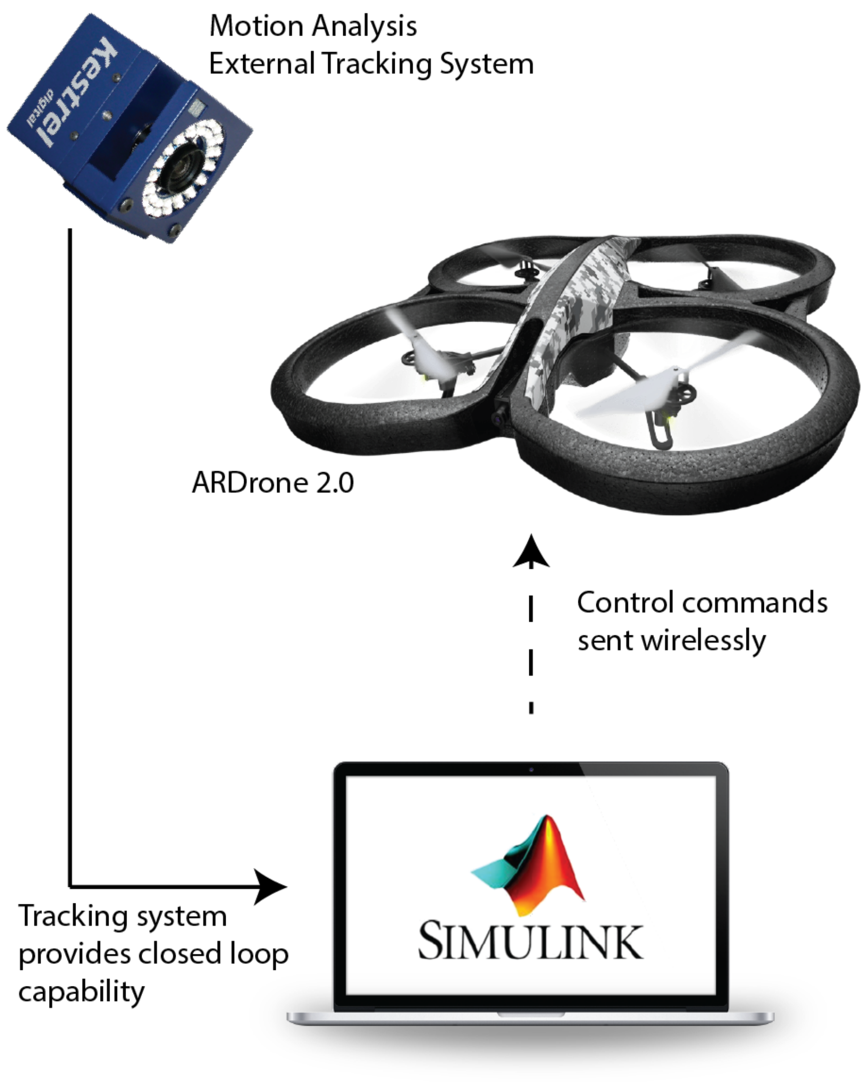

Motion Analysis
External Tracking System
Kestrel digital
ARDrone 2.0
Control commands sent wirelessly
Tracking system provides closed loop capability
SIMULINK

Figure 6.4: The experimental setup for this work includes a Motion Analysis external tracking system with 16 cameras which provides the position of the drone and obstacles in the environment. The same Simulink model used in Chapter 5 provides control commands to the ARDrone over a wireless connection.

6.2 Experimental Results and Discussion

Having validated the controller using the Simulink simulation, it was then implemented on the actual ARDrone. Several experiments were formed, in the order outlined in Table 6.1.

Table 6.1: Experimental Tests

Test Number	**Target**	**Obstacle(s)**
1	1 - Static	0 - N/A
2	1 - Static	1 - Static
3	1 - Dynamic Square	0 - N/A
4	4 - Static Waypoints	1 - Static
5	1 - Dynamic No Pattern	0 - N/A

Because the simulation showed a clear improvement in performance between the tradition PFC and the developed ePFC, the traditional controller was not tested on the experimental platform. Instead, the ePFC was immediately implemented in the experiments.

In the first test, the drone was placed approximately 4.2m from the target's location. Because the target wand is often held by a human, the drone was requested to fly to 1m away from the target location to avoid collision with someone holding the wand. The drone's response shown in Fig. 6.5 demonstrates the capability of the drone to achieve a goal position effectively. Starting at approximately 7.75sec, the drone enters an autonomous mode, and achieves stable hover 1m away from the target in approximately 5sec. It is important to note that while the drone did overshoot it's goal location, it did not overshoot enough to get close to hitting the target. The closest that the drone got to the target was just under 0.75m.

In the second experiment, the drone was placed approximately 5.2m away from the target wand, and an obstacle was located in the path lying directly between the drone and the target. Similar to the first test, the drone's mission was to fly to within 1m of

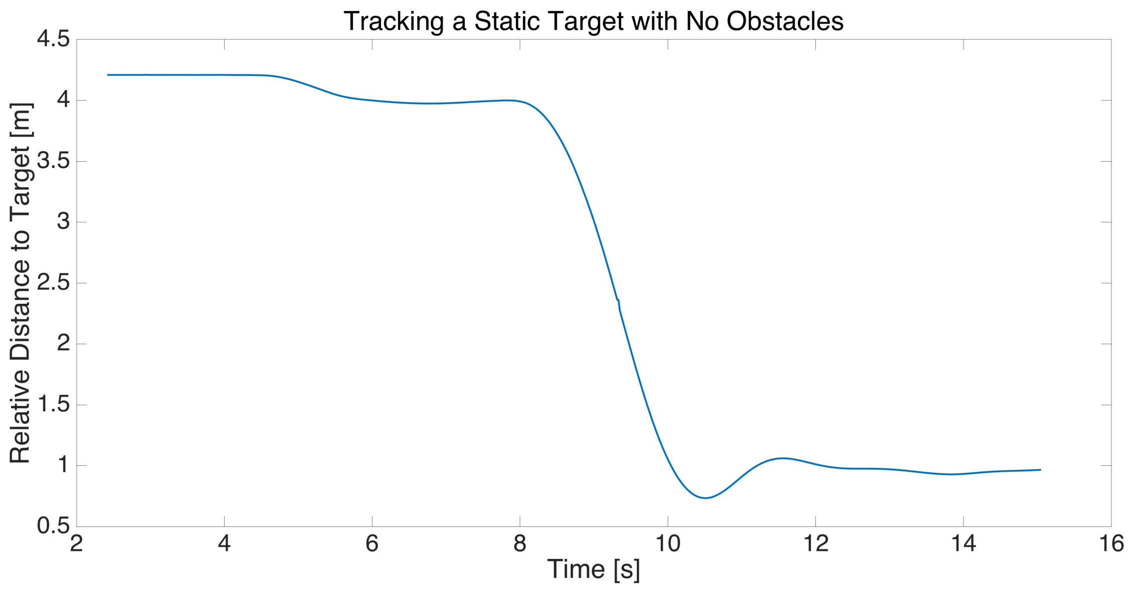


Figure 6.5: The results of tracking a static target with no obstacles are very good. With an initial condition of approximately 3.2m, the drone achieves position in under 5sec.

the target, this time while avoiding the obstacle and still achieving the task. As the drone begins moving towards the target, it also moves towards the obstacle. Because of the repulsive forces generated by the relative position and velocity with respect to the obstacle, the drone is elegantly pushed around the obstacle and still makes it to the target location. Figure 6.6 shows the results of this test, demonstrating that the drone maintains a safe distance from the obstacle (1m minimum) and also achieves the goal.

In the third test, the drone was instructed to follow the target wand as it moved in an approximate rectangle around the lab. The results shown in Fig. 6.7 illustrate the path of the drone as it follows the target through the pattern. As shown, the drone does in fact track the rectangle as instructed. Because the path of the target was moved manually by a person holding the wand, the target trajectory is not a perfect rectangle. Therefore, the next experiment establishes a perfect rectangle using virtual waypoints.

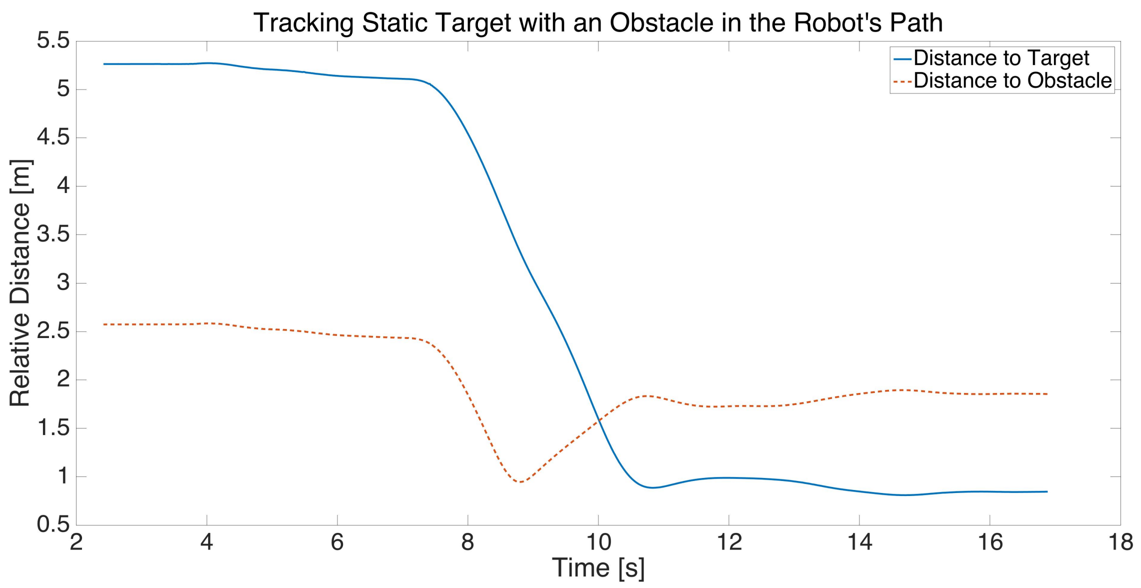


Figure 6.6: The results of tracking a static target with an obstacle in the way demonstrates the controllers effectiveness at avoiding collisions. As expected, the drone's position relative to the obstacle decreases, but the drone takes avoidance action and never gets closer than one meter away from the obstacle.

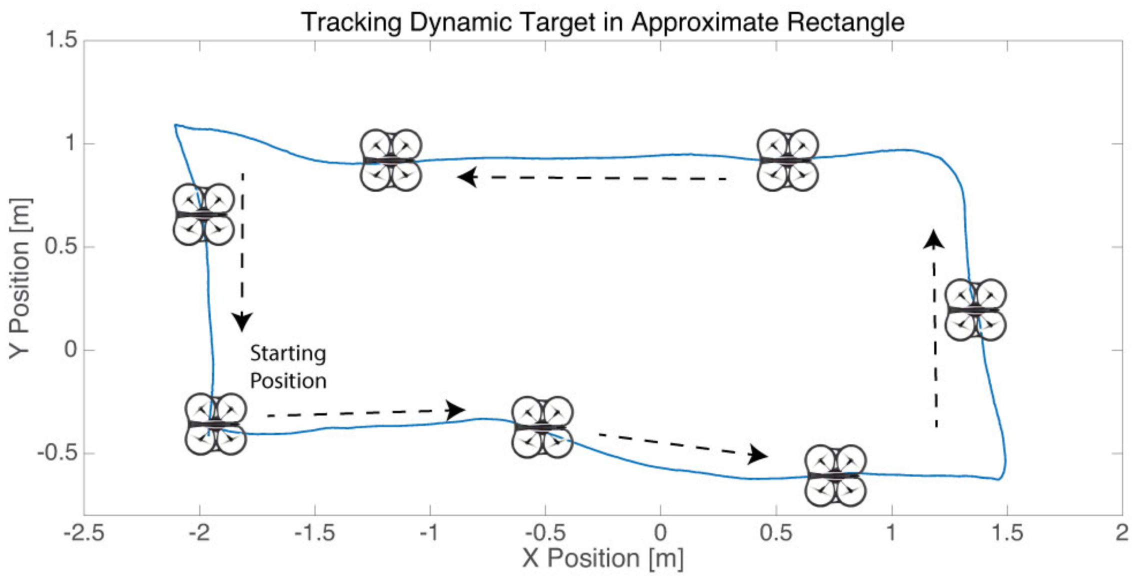


Figure 6.7: In the third experiment, the drone tracks a target which moves in an approximate rectangle. As shown, the drone does track, but because the desired trajectory is human-controlled the reference is not perfect. Test number four addresses this imperfection by using set virtual waypoints.

In test number four, virtual waypoints like those used in simulation are used to demonstrate the ability of the drone to navigate a course and avoid obstacles. The waypoints used for this test are outlined in Table 6.2.

Table 6.2: Experimental Waypoints

Waypoint	**X Coordinate [m]**	**Y Coordinate [m]**
1	1.5	-0.5
2	1.5	0.5
3	-1.5	0.5
4	-1.5	-0.5

The results from the fourth experiment shown in Fig. 6.8 and Fig. 6.9 demonstrate that the drone successfully reaches each waypoint, and also avoids the obstacle in its path between waypoints two and three.

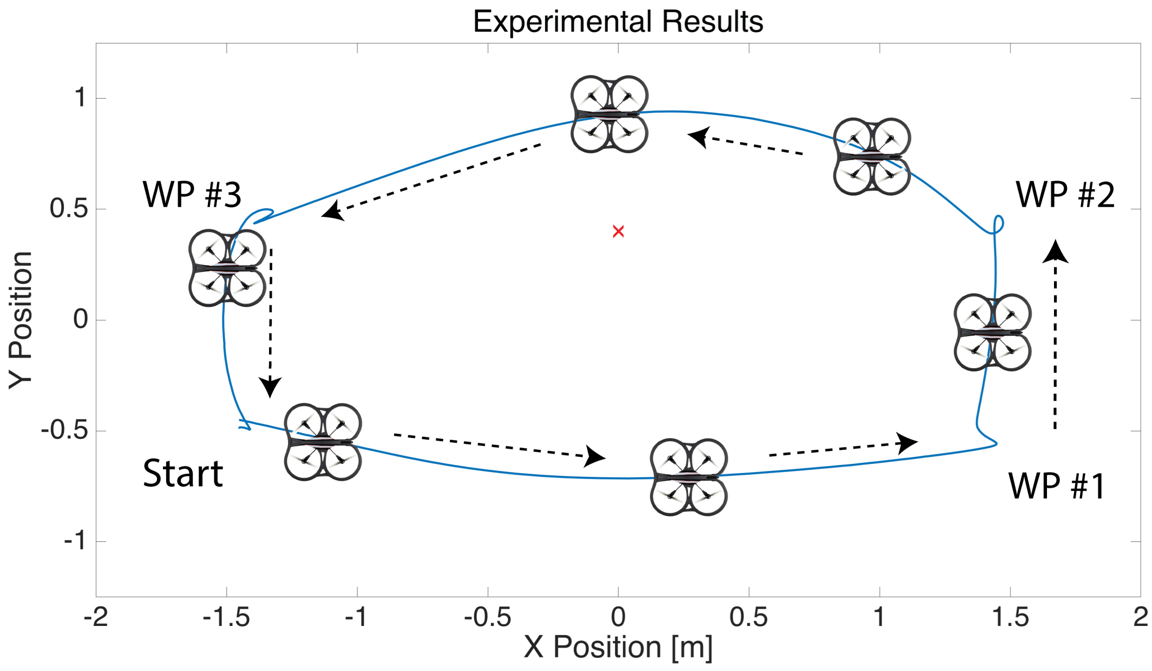


Figure 6.8: To test the ePFC experimentally, the drone follows waypoints similar to those in the simulation. The drone successfully reaches each waypoint and avoids the obstacle in its path between waypoints two and three.

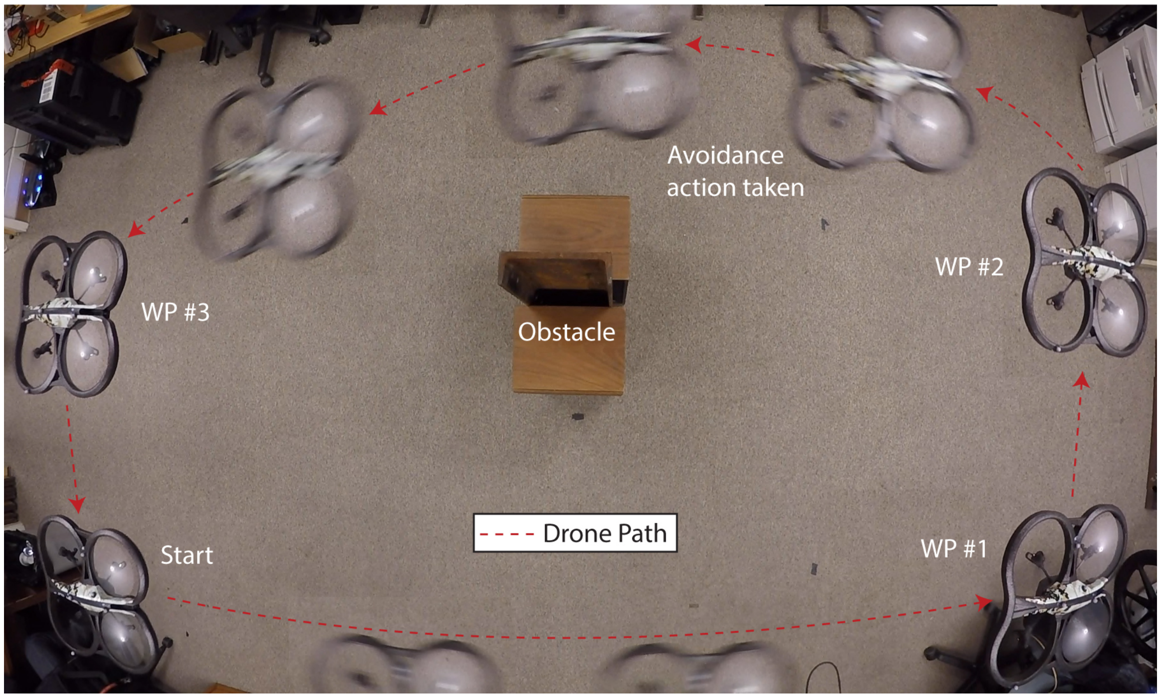


Figure 6.9: Stills from a video demonstrate the drone taking avoidance action while tracking moving waypoints.

To quantify the controller performance, the error in response to a waypoint change, or step input, is shown in Fig. 6.10. The X axis error is chosen as the worst case scenario in the experiment, having a step input of over 2.5m versus only 1m on the Y axis.

The controller's performance is quite good to step inputs, with an approximate settling time of 5.5sec, and a percent overshoot of only 1.8%. A comparison between the simulated and experimental results is outlined in Table 6.3. While the experimental results do have a slightly longer settling time, and more overshoot, this is not surprising. In a real world application, the controller is subject to disturbances such as ground effects from propeller wash, since the drone is operating close to the ground and desks.

As the drone approaches the obstacle, the repulsive potential pushes the drone around it as expected. In this experiment, the drone avoids the obstacle by a margin

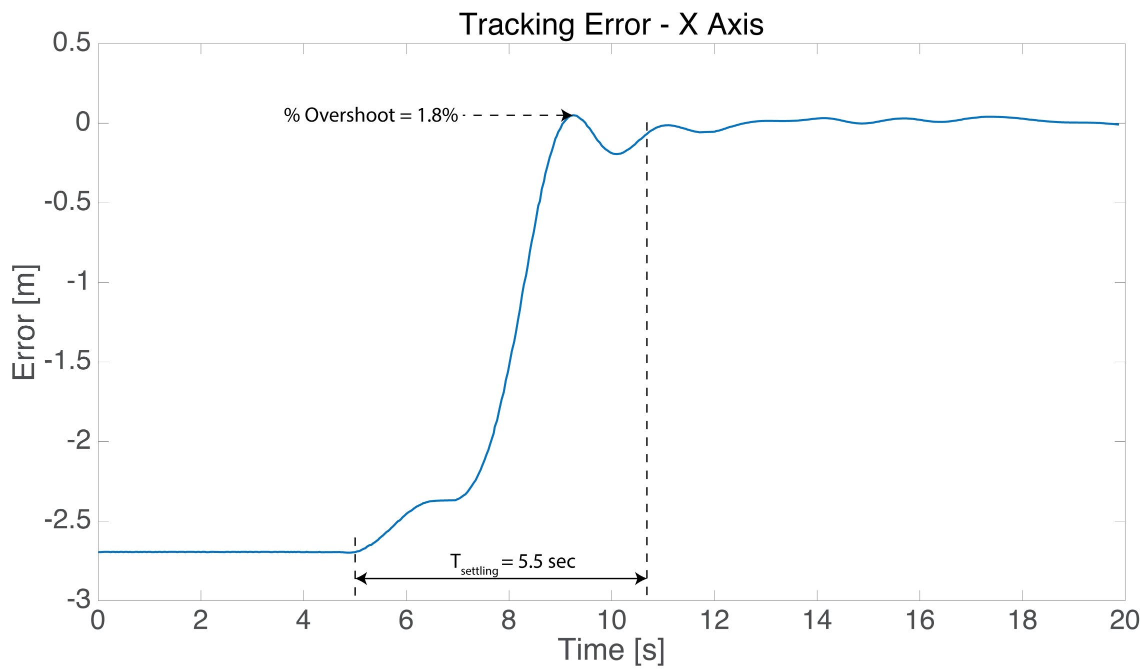


Figure 6.10: The error in response to a waypoint change, or a step input, results in a settling time of approximately 5.5sec and a percent overshoot of only 1.8%. The X axis was chosen because the step input for this direction was the largest, at over 2.5m, whereas the Y input is only 1m.

Table 6.3: Simulation vs Experimental Evaluation

Experiment	**Overshoot** [%]	**Settling Time** [sec]
Simulated ePFC	0%	5
Experimental ePFC	1.8%	5.5

of approximately 0.5m. Thus, this demonstrates that the drone can successfully avoid obstacles.

In addition to tracking static targets and virtual waypoints, a final test is performed in which the ARDrone is commanded to follow the target as it moves about the lab environment in an arbitrary pattern. During this experiment, the drone must maintain a safe distance at all times and should always face the target. This task was performed several times to evaluate the performance. In each of the tests the drone successfully completes the task. Even under extreme circumstances (e.g., very fast maneuvers) the drone is able to recover and maintain the desired behavior. Figure 6.11 shows frames from a video [49] taken of the drone performing this task. In the video it can clearly be seen that the drone follows the target around while always maintaining the proper heading to face the target.

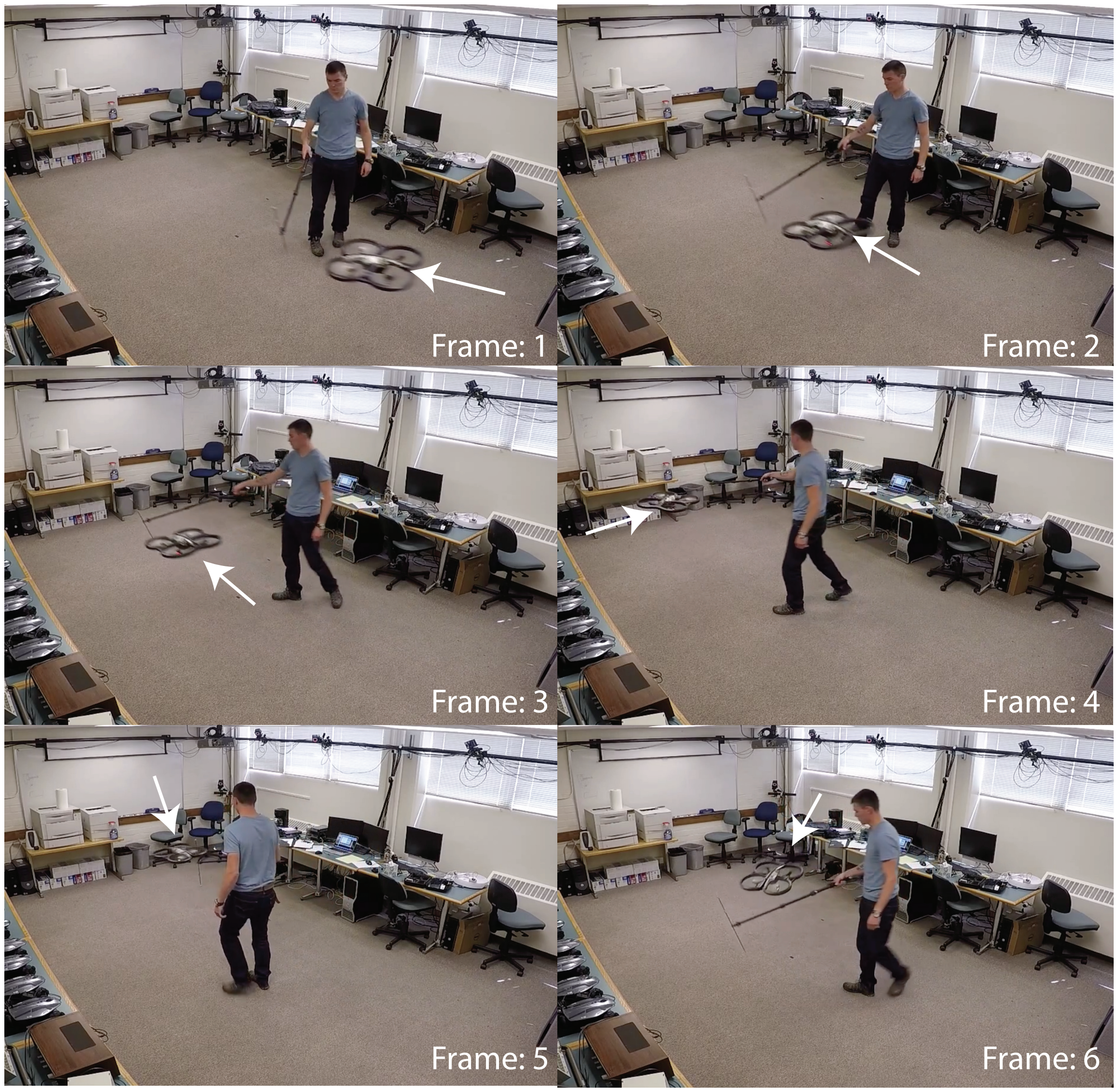


Figure 6.11: The third experiment demonstrates the drone tracking a dynamic target, while maintaining the proper heading to always face the target.

6.3 Summary

This chapter presented the experimental setup used to implement the proposed controller as well as experimental results obtained. The results demonstrate that the drone is able to successfully track both static and dynamic targets, as well as avoid obstacles in its path.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis presented an extended potential field controller (ePFC) which augments the traditional PFC with the capability to use relative velocities between a drone and a target or obstacles as feedback for control. Next, the stability of the ePFC was proven using Lyapunov methods. Additionally, the presented controller was simulated and its performance relative to a tradition PFC was evaluated. The evaluation shows that the ePFC performs significantly better than a traditional PFC by reducing overshoot and settling time when navigating between waypoints. Finally, experimental results were presented which showed the actual performance of the proposed controller.

7.2 Future Work

Future work may include using an experimental system with completely onboard sensing capabilities. Potentially, the front facing camera on the ARDrone 2.0 could be used for localization using computer vision algorithms. Additionally, the controller may be implemented onboard the drone itself. Because the ePFC presented is not computationally intensive, it can be implemented on nearly any drone on the market today.

Bibliography

[1] Motion analysis systems. [Online]. Available: http://www.motionanalysis.com

[2] J. M. de Dios, L. Merino, F. Caballero, A. Ollero, and D. Viegas, "Experimental results of automatic fire detection and monitoring with uavs," *Forest Ecology and Management*, vol. 234, Supplement, pp. S232 –, 2006.

[3] L. Merino, F. Caballero, J. Martnez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple uavs: Application to automatic detection of forest fires," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 165–184, 2006.

[4] D. Casbeer, R. Beard, T. McLain, S.-M. Li, and R. Mehra, "Forest fire monitoring with multiple small uavs," in *American Control Conference, 2005. Proceedings of the 2005*, June 2005, pp. 3530–3535 vol. 5.

[5] P. Sujit, D. Kingston, and R. Beard, "Cooperative forest fire monitoring using multiple uavs," in *Decision and Control, 2007 46th IEEE Conference on*, Dec 2007, pp. 4875–4880.

[6] C. Yuan, Z. Liu, and Y. Zhang, "Uav-based forest fire detection and tracking using image processing techniques," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, June 2015, pp. 639–643.

[7] R. Pitre, X. Li, and R. Delbalzo, “Uav route planning for joint search and track missions: An information-value approach,” *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 48, no. 3, pp. 2551–2565, July 2012.

[8] A. Birk, B. Wiggerich, H. Bulow, M. Pfingsthorn, and S. Schwertfeger, “Safety, security, and rescue missions with an unmanned aerial vehicle (uav),” *Journal of Intelligent and Robotic Systems*, vol. 64, no. 1, pp. 57–76, 2011.

[9] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, “Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue,” *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 46–56, Sept 2012.

[10] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, “Supporting wilderness search and rescue using a camera-equipped mini uav,” *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 89–110, 2008.

[11] D. Erdos, A. Erdos, and S. Watkins, “An experimental uav system for search and rescue challenge,” *Aerospace and Electronic Systems Magazine, IEEE*, vol. 28, no. 5, pp. 32–37, May 2013.

[12] M. Quaritsch, K. Kruggl, D. Wischounig-Strucl, S. Bhattacharya, M. Shah, and B. Rinner, “Networked uavs as aerial sensor network for disaster management applications,” *Elektrotechnik und Informationstechnik*, vol. 127, no. 3, pp. 56–63, 2010.

[13] I. Maza, F. Caballero, J. Capitan, J. Martinez-de Dios, and A. Ollero, “Experimental results in multi-uav coordination for disaster management and civil

security applications," *Journal of Intelligent and Robotic Systems*, vol. 61, no. 1-4, pp. 563–585, 2011.

[14] P. Bupe, R. Haddad, and F. Rios-Gutierrez, "Relief and emergency communication network based on an autonomous decentralized uav clustering network," in *SoutheastCon 2015*, April 2015, pp. 1–8.

[15] H. La, R. Lim, B. Basily, N. Gucunski, J. Yi, A. Maher, F. Romero, and H. Parvardeh, "Mechatronic systems design for an autonomous robotic system for high-efficiency bridge deck inspection and evaluation," *Mechatronics, IEEE/ASME Transactions on*, vol. 18, no. 6, pp. 1655–1664, Dec 2013.

[16] S. J. Mills, J. J. Ford, and L. Mejias, "Vision based control for fixed wing uavs inspecting locally linear infrastructure using skid-to-turn maneuvers," *Journal of Intelligent and Robotic Systems*, vol. 61, no. 1-4, pp. 29–42, 2011.

[17] Z. Li, Y. Liu, R. Walker, R. Hayward, and J. Zhang, "Towards automatic power line detection for a uav surveillance system using pulse coupled neural filter and an improved hough transform," *Machine Vision and Applications*, vol. 21, no. 5, pp. 677–686, 2010.

[18] R. Bloss, "Unmanned vehicles while becoming smaller and smarter are addressing new applications in medical, agriculture, in addition to military and security," *Industrial Robot: An International Journal*, vol. 41, no. 1, pp. 82–86, 2014.

[19] J. Roldan, G. Joossen, D. Sanz, J. del Cerro, and A. Barrientos, "Mini-uav based sensory system for measuring environmental variables in greenhouses," *Sensors*, vol. 15, no. 2, pp. 3334–3350, 2015.

[20] M. Rossi, D. Brunelli, A. Adami, L. Lorenzelli, F. Menna, and F. Remondino, "Gas-drone: Portable gas sensing system on uavs for gas leakage localization," in *SENSORS, 2014 IEEE*, Nov 2014, pp. 1431–1434.

[21] Hokuyo. [Online]. Available: https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html

[22] Y. Okubo, C. Ye, and J. Borenstein, "Characterization of the Hokuyo URG-04LX laser rangefinder for mobile robot obstacle negotiation," vol. 7332, 2009.

[23] M. Shaohua, X. Jinwu, and L. Zhangping, "Navigation of micro aerial vehicle in unknown environments," in *25th Chinese Control and Decision Conference (CCDC)*, 2013, pp. 322–327.

[24] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, 2012.

[25] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained mav," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 20–25.

[26] I. Sa and P. Corke, "System identification, estimation and control for a cost effective open-source quadcopter," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 2202–2209.

[27] A. Kendall, N. Salvapantula, and K. Stol, "On-board object tracking control of a quadcopter with monocular vision," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014, pp. 404–411.

[28] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3d exploration with a micro-aerial vehicle," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 9–15.

[29] J.-E. Gomez-Balderas, P. Castillo, J. Guerrero, and R. Lozano, “Vision based tracking for a quadrotor using vanishing points,” *Journal of Intelligent and Robotic Systems*, vol. 65, no. 1-4, pp. 361–371, 2012.

[30] L. Meier, P. Tanskanen, L. Heng, G. Lee, F. Fraundorfer, and M. Pollefeys, “Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision,” *Autonomous Robots*, vol. 33, no. 1-2, pp. 21–39, 2012.

[31] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[32] P. Eklund, S. Kirkby, and S. Pollitt, “A dynamic multi-source dijkstra’s algorithm for vehicle routing,” in *Intelligent Information Systems, 1996., Australian and New Zealand Conference on*, Nov 1996, pp. 329–333.

[33] M. Parulekar, V. Padte, T. Shah, K. Shroff, and R. Shetty, “Automatic vehicle navigation using dijkstra’s algorithm,” in *Advances in Technology and Engineering (ICATE), 2013 International Conference on*, Jan 2013, pp. 1–5.

[34] Z. Yan and Z. Jun, “Dijkstra’s algorithm based robust optimization to airline network planning,” in *Mechanic Automation and Control Engineering (MACE), 2010 International Conference on*, June 2010, pp. 2783–2786.

[35] P. Hart, N. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, July 1968.

[36] W. Zeng and R. L. Church, “Finding shortest paths on real road networks: the case for a*,” *International Journal of Geographical Information Science*, vol. 23, no. 4, pp. 531–543, 2009.

[37] W. G. Walter, “An imitation of artificial life,” *Scientific American*, vol. 96, pp. 123–137, 1950.

[38] ——, “An electro-mechanical animal ,” *Dialectica*, vol. 4, no. 3, pp. 206–213, 1950.

[39] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 2520–2525.

[40] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, *IEEE Robotics Automation Magazine.*

[41] A. Dogan, “Probabilistic approach in path planning for uavs,” in *Intelligent Control. 2003 IEEE International Symposium on*, Oct 2003, pp. 608–613.

[42] Y. Kuwata, T. Schouwenaars, A. Richards, and J. How, “Robust constrained receding horizon control for trajectory planning,” in *Proceedings of the AIAA guidance, navigation and control conference*, 2005.

[43] A. A. Neto, D. Macharet, and M. Campos, “Feasible path planning for fixed-wing uavs using seventh order bezier curves,” *Journal of the Brazilian Computer Society*, vol. 19, no. 2, pp. 193–203, 2013.

[44] A. Altmann, M. Niendorf, M. Bednar, and R. Reichel, “Improved 3d interpolation-based path planning for a fixed-wing unmanned aircraft,” *Journal of Intelligent and Robotic Systems*, vol. 76, no. 1, pp. 185–197, 2014.

[45] H. M. La, R. S. Lim, W. Sheng, and J. Chen, “Cooperative flocking and learning in multi-robot systems for predator avoidance,” in *IEEE 3rd Annual International Conference on Cyber Technology in Automation, Control and Intelligent Systems (CYBER)*, May 2013, pp. 337–342.

[46] H. M. La and W. Sheng, "Multi-agent motion control in cluttered and noisy environments," *Journal of Communications*, vol. 8, no. 1, p. 32, 2013.

[47] ——, "Dynamic target tracking and observing in a mobile sensor network," *Robotics and Autonomous Systems*, vol. 60, no. 7, p. 996, 2012.

[48] D. E. Sanabria. Ardrone simulink development kit v1.1. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/43719-ar-drone-simulink-development-kit-v1-1

[49] A. C. Woods. Dynamic target tracking with ardrone. [Online]. Available: https://youtu.be/v85hs8-uc1s