Haptic Collision Avoidance for a Remotely Operated Quadrotor UAV in Indoor Environments

Adam Brandt

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Mark B Colton, Chair
Timothy W. McLain
Randal W. Beard

Department of Mechanical Engineering

Brigham Young University

December 2009

ABSTRACT

Haptic Collision Avoidance for a Remotely Operated Quadrotor

UAV in Indoor Environments

Adam M. Brandt

Department of Mechanical Engineering

Master of Science

A quadrotor is an omnidirectional unmanned air vehicle that is suitable for indoor flight because of its ability to hover and maneuver in confined spaces. The remote operation of this type of vehicle is difficult due to a lack of sensory perception; typically, the view from the onboard camera is the only information transmitted to the pilot. This thesis proposes using force feedback exerted by the command input device on the hand of the pilot to assist in avoiding collisions while navigating in indoor environments.

Five candidate algorithms are presented for calculating the forces to be felt by the pilot based on the quadrotor's position and velocity in the indoor environment. The candidates include a parametric algorithm based on the dynamics of the quadrotor, two time-to-impact algorithms, and two algorithms that employ virtual springs between the quadrotor and obstacles. A method of incorporating the position of the command input device to improve the usability and effectiveness of the algorithms is also presented.

A framework for simulating the quadrotor dynamics, indoor environment, and force feedback algorithms is described. In the simulation, the pilot commands a simulated quadrotor, using a commercial haptic interface, as it flies in an indoor environment. The pilot receives force feedback cues as the quadrotor navigates around obstacles. Two methods of control were used for the simulation. In the first method, displacements of the haptic interface correspond to velocity commands to the quadrotor. In the second method, displacements of the input correspond to desired roll and pitch commands.

Two user study experiments, one for each control method, were performed to compare the force feedback algorithms in simulation. The results of the velocity control experiment suggest that higher force levels help to avoid collisions and that the time to impact algorithm results in fewer collisions than having no force, but is not significantly better than the other algorithms. The results of the angle control experiment suggest that the time to impact algorithm

is clearly the best in terms of hits and hit length and has no disadvantages compared to the other algorithms.

Finally, to demonstrate the force feedback algorithms and software in a real-world environment, the system was interfaced with a physical quadrotor. The quadrotor system is described and the results of the tests are presented.

Keywords: Adam Brandt, haptic, quadrotor, UAV, collision avoidance, force feedback

ACKNOWLEDGMENTS

Many people have been very influential in helping me to complete my thesis and my masters degree. Among these are professors, colleagues, and family, many of whom I'm not sure I could have completed this without.

First I would like to acknowledge the continual support help, and love that I have received from my wife Jenn and my son Collin. They have always been there for me in anything I needed and understood the hours I needed to spend away from home. I would also acknowledge other family who took an interest in what I was doing and listened to me explain things that probably didn't make any sense.

My advisor, Dr. Colton, has been a tremendous help in guiding me through my masters degree. He supplied the concept for my thesis and always made himself available to listen to my problems, give me advice, and crash the quadrotor in my simulator. I would also like to recognize Dr. Beard, who let me use his quadrotor simulation to help jumpstart my own, and Dr. McLain, who helped me better understand the dynamics and control of aircraft.

The MAGICC Lab has been a great place to work. Everyone was always open to sharing ideas and helping each other. Among my colleagues in the MAGICC Lab I would like to acknowledge Dan Carlson and Dan Ricks who sat on either side of me and were always willing to give advice and be guinea pigs for my simulations. Travis Brown provided a great service to me in developing a library so that I could communicate with the PHANTOM in Matlab. Most recently, Caleb Chamberlain has gone out of his way to help me communicate with the quadrotor and get it flying so that I could test out the force feedback.

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

Quadrotors are a type of rotorcraft that employ four coplanar rotors to generate thrust for hovering and maneuvering. The rotors are typically positioned symmetrically about a central body, which houses the battery, autopilot and other computing hardware. A camera can be placed on the quadrotor to send live video to the pilot for remote operations. Figure 1-1 shows a typical quadrotor.

Figure 1-1: A typical quadrotor with four coplanar rotors around the main body that houses the battery and computing hardware. This quadrotor does not have a camera.

Quadrotors have two primary advantages over fixed-wing aircraft for certain unmanned aerial vehicle (UAV) applications. First, they have the ability to hover, which is beneficial when it is desired to remain stationary, as in certain reconnaissance and search and rescue missions. Second, quadrotors are omni-directional, allowing them to move freely in any direction. This is typically not true for fixed-wing aircraft because they generally require a forward velocity to create lift. The hover and omni-directional capabilities of quadrotors make them a popular choice for applications in indoor environments.

Because the quadrotor can hover and even rotate in place, it has the ability to scan a room or hallway. The fact that it can move in any direction makes it well suited for navigating tight areas and avoiding obstacles. At the same time, the ability to move in any direction means that it can move in directions that are not within the camera's field of view. This makes the quadrotor susceptible to collisions.

The purpose of this thesis is to explore one possible solution to this problem: the use of force feedback to give the pilot haptic (touch) information about the quadrotor's environment. The objective of force feedback is to provide additional information to a remote pilot via the sense of touch to assist in obstacle avoidance in indoor environments. When the quadrotor sensors indicate that it is approaching an obstacle the input device exerts appropriate forces on the hand of the pilot to help him or her to avoid a collision.

1.1 Motivation

Operating a vehicle from a remote location is desirable for many reasons. First, the vehicle can be smaller, lighter, and more discreet because it does not require a human occupant.

This generally makes the vehicle less expensive as well. Second, when the vehicle must pass through dangerous or hazardous areas, there is no possibility for harm to the operator.

One example of a situation in which remote operation of a vehicle would be appropriate is in a search and rescue application. For example, a remote vehicle could search a building that has been damaged in a natural disaster or where hazardous material has been spilt. The vehicle could be used to search for survivors in the building, to evaluate the condition of the building, to create a plan for recovery, or to determine if the building is safe for people to enter.

1.1.1 Quadrotors

The two possible vehicle types that could be used for searching a building are ground vehicles and air vehicles. A ground vehicle may work well for searching buildings in which the floors are level and free from debris. However, in the presence of obstacles, debris, or stairs, a ground vehicle may be less effective than an air vehicle. For indoor flight, rotorcraft are the most appropriate type of air vehicle because of their ability to hover and make sharp turns. Because fixed-wing aircraft require airspeed to maintain lift, they do not have these same abilities. For the present work, a quadrotor was chosen over a helicopter as the rotorcraft of choice because of the simpler design, dynamics, and control.

1.1.2 Force Feedback

One of the most important problems with the remote operation of vehicles is that sensory perception is lost. In most cases the only information that is transmitted to the remote pilot from the vehicle is a video feed from a camera placed on the vehicle. Outdoor vehicles also have the ability to use and provide position information through GPS, but this is not possible in indoor environments.

The visibility from a remote camera has been described as “looking at the world through a soda straw” [1]. This makes it difficult for the operator to obtain sufficient knowledge of the surrounding of the quadrotor. This decrease in situational awareness could result in an increased time to complete a task, increased workload on the human operator, decreased mission effectiveness, or collisions with obstacles within the environment. One potential solution is to rotate the entire vehicle to allow the operator to view the entire environment, but this is inconvenient and inefficient, and other views are lost while rotating. Another option is to have a gimbaled camera, but this adds complexity and weight to the quadrotor, as well as requiring more power and cost. Instead, there could be multiple cameras, but this also adds complexity and possible bandwidth issues, and the remote pilot would have to take in multiple views. It is important for the remote operator of a quadrotor to know if there are potential hazards to the sides or behind of the vehicle, or even above and below it, because it has the capability to move in all those directions.

Force feedback, given through the controller to the hand of the user, would give the operator information about the environment surrounding the quadrotor naturally, through the sense of touch. As a remotely operated vehicle travels toward an obstacle or wall, the controller exerts forces on the pilot’s hand in a direction away from the obstacle or wall. This force would not overpower the user, but serve as a warning to help avoid hitting the obstacle. This is the proposed method for this thesis to give the pilot more sensory information and avoid collisions.

1.2 Related Work

Force feedback has been explored by other researchers to assist in collision avoidance and in other guidance applications. These applications include direct vehicle operation,

teleoperation of robot manipulators, remote operation of ground robots, and remote operation of UAVs.

1.2.1 Vehicle Operation

The use of haptic guidance for vehicle operation is different from the other uses that will be discussed because the driver is inside the vehicle and not remotely operating it. Although in this case the purpose of force feedback is not to make up for the loss of sensory perception, it is still similar in the fact that the forces are there to assist the driver to achieve a goal.

Two types of haptic feedback for vehicle operation have been explored. The first is haptic feedback on the gas pedal to assist drivers in keeping safe distances when following other cars. Mulder et al. [2] have done extensive research in this area, finding that, depending on the type of feedback, haptic feedback was successful in increasing car-following performance and allowing the driver to have more time for other processes. The second application of haptic feedback for vehicle operation is feedback on the steering wheel to support steering tasks. This work was also accomplished by Mulder et al. [3], who performed experiments with drivers navigating turns with and without haptic feedback. These experiments found that haptic feedback significantly improved the safety boundaries and made steering activity smoother and less active. Both of the applications showed that haptic feedback is promising in assisting drivers who are directly operating vehicles.

1.2.2 Robot Manipulators

Haptic feedback to assist in operation of robot manipulators is also different from the work presented in this thesis in that the operator is not guiding a moving vehicle, but rather a robot arm on a remote stationary platform. One example of this is attributed to Yano et al. [4]

who used haptic feedback to assist crane operators. A special joystick was used to send suggestive information to the operator using haptic feedback. This gave information such as gravity compensation and restriction on the crane's velocity, as well as collision avoidance feedback. They found through experimentation that their system did allow an operator to transfer a load quickly without hitting obstacles. Another example of this is research by Frey et al. [5] on using a full arm force feedback device to help users move a simulated robot arm to a desired position and apply a force. They performed a user study which suggested that the force feedback allowed users to reach the desired position more quickly and more easily generate the required forces.

Another application of haptic feedback for robot manipulators is research on control of a dual arm humanoid robot done by Charoenseang et al. [6]. The authors here used an inexpensive commercial joystick to control the arms of robot in simulation. The system was designed for robot teleassistance applications. In their experiments, the users received visual information as well as the force feedback information. When the simulator detected a possible collision it would send haptic feedback to the user through the joystick. To detect collisions the authors created three virtual spheres around the elbow, wrist, and end-effector of the robot arm. They found that this was a simple and effective way to help the robot to avoid collisions with other objects and with its other arm.

1.2.3 Ground Robots

Most research on haptic collision avoidance has been related to unmanned ground vehicles (UGVs), which are remotely controlled or autonomous mobile robots. Borenstein and Koren [7] used a vector field histogram (VFH) method to compute forces to help the user avoid collisions, while at the same time guiding the vehicle towards a target. They found that their

VHF method was robust and allowed fast movement of the robot without it stopping in front of obstacles. Similarly, Lee et al. [8] performed a user study with ground robots using sensors to find range information and converting the range information into forces displayed to the user through a haptic device. Their user study found that the haptic feedback increased operator performance and reduced collisions without significantly increasing navigation time.

Hong et al. [9] demonstrated the use of haptic feedback with a robot capable of climbing stairs. The haptic feedback not only assisted in collision avoidance, but also in stair climbing as the forces helped the user to feel when a leg would touch the ground. Barnes and Counsell [10] explored haptic feedback collision avoidance, with the user having full control of the ground robot as well as the robot having semi-autonomous control. The results of their user study showed that operator performance improved when haptic information was given when the user had full control, but it was inconclusive in the semi-autonomous control mode.

1.2.4 UAVs

UAVs differ from UGVs in certain key areas. First, UAVs have a greater number of degrees of freedom, thus requiring more control and more possibility for collisions. Second, the load-bearing capabilities of UAVs are much lower, and the amount of equipment and computational power is therefore correspondingly decreased. As a result, the sensing capabilities of UAVs are, in general, also more restricted. Previous work with UGVs often made use of systems with multiple sensors, which is not realistic for most small air vehicles. Decreased sensing capabilities results in a lower-fidelity model of the robot's surroundings, making it more challenging to generate feedback forces for the pilot.

Much of the work done in haptic collision avoidance for UAVs has been done by Lam, Mulder, and van Paassen at the Delft University of Technology in the Netherlands. Their earlier

research focused on development of artificial risk fields used to calculate the forces felt by the operator when approaching potential hazards. More recently they have researched how to deal with time delays [11] and degraded visual interfaces [12].

In 2004 Boschloo et al. [13] published an article that reviewed two types of risk fields with two types of force vector calculations applied to each. They tested a basic risk field, which has forces increasing exponentially as the vehicle approaches an obstacle, and a parametric risk field, where the forces increase according to a more complex function in zones that are determined by different parameters, using radial and normal risk vectors. The simulation was done in two dimensions with no hardware or human in the loop. They found that the parametric risk field with radial risk vectors produced the fewest collisions in the simulations.

This research was continued by Lam et al. [14], who performed a user study on the two risk fields. In this simulation the user had a three dimensional view of the world, but motion was limited to translation in two dimensions. The force feedback was provided by an electro-hydraulic side-stick and the simulation was done in a fixed-base part-task simulator. The results of the experiment were quantified by number of collisions, elapsed time, average speed, workload using NASA-TLX [15], and a user questionnaire. The experiment showed that the haptic feedback did help to avoid collisions, but sometimes required a higher workload when the repulsive forces from the haptic control device required counteracting forces from the operator.

1.3 Contributions

This thesis adds to work done previously by developing and testing new algorithms, employing new concepts on which the algorithms are based. Also, the algorithms are developed for three dimensions and the simulation is extended into three dimensions by using a haptic

device that is capable of giving force feedback in three dimensions. Another aspect of the simulation that contributes to this research is that the force feedback is based off of limited range information, as if from real sensors placed on the quadrotor, rather than a perfect knowledge of the surroundings. The system and algorithms are demonstrated by piloting a real UAV in an indoor environment, which takes the work beyond the simulations presented in previous work.

1.4 Thesis Overview

This thesis will present novel force feedback algorithms designed to assist a pilot in maneuvering a quadrotor through an indoor environment. The simulation that was used to test the algorithms in a virtual environment will be described, along with the haptic interface used by the pilot to generate command inputs and receive force feedback cues from the simulation. Two human subject experiments to test the effectiveness of these algorithms will be described. Application of force feedback with a real quadrotor will be demonstrated. Finally, conclusions will be made about the effectiveness of the system and possible future work and applications.

2 Force Feedback Algorithms

Force feedback can help a remote pilot to avoid collisions, but if the forces are not properly designed they can be at best unhelpful, and possibly even harmful. Five candidate force feedback algorithms were developed for comparison against each other and against no force feedback. This section describes the algorithms, which include:

- Dynamic Parametric Field
- Time to Impact
- Modified Time to Impact
- Spring Algorithm
- Spring Algorithm with velocity dependence

2.1 Dynamic Parametric Field

The Dynamic Parametric Field (DPF) is modeled after the concept of the parameterized risk field, developed by Boschloo et al. [13], of having different zones defined by calculated distances. Their field uses a minimum stopping distance calculated from the velocity and the maximum deceleration of the vehicle. This distance defines the critical region where the risk is greatest. Another distance defines the beginning of the risk field which is calculated by multiplying the velocity by a time constant. The problem with using these calculations to define the distances is that the quadrotor does not instantly reach its maximum deceleration and hold

that deceleration until it stops. Also, there is no signal for the user to tell the difference between the normal risk area and the critical area. The Dynamic Parametric Field uses different calculations to determine distances of the zones and has differently defined zones.

The dynamic capabilities of the quadrotor were accounted for when developing the distances in the Dynamic Parametric Field. As the quadrotor approaches an obstacle, it passes through four zones that are used to determine the forces applied to the pilot's hand, as shown in Figure 2-1. In this figure, *d* is the distance of the quadrotor from the wall. The distance from the wall that forces begin to be felt *is* d_{max}, $d_{desired}$ is the desired stopping distance for a given velocity, d_{mid} is the beginning of the transition zone, and d_{min} is the minimum required stopping distance for a given velocity. Two of the distances, d_{min} and $d_{desired,}$ were determined by simulating the autopilot and quadrotor dynamics with initial velocities and constant desired velocity inputs. The other distances are tuned by the user. A less experienced user may choose higher values for the parameters to give longer warning distances and more force, whereas a more experienced user may choose lower values for the parameters to receive less force feedback assistance. A description of the various zones in Figure 2-1 will now be presented.

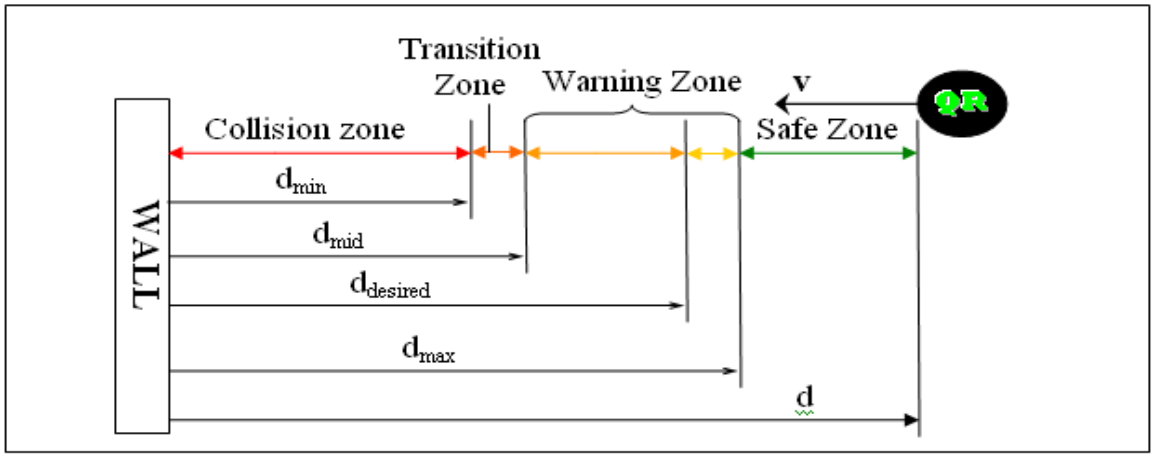

Figure 2-1: Dynamic parametric field schematic

2.1.1 Collision Zone

The collision zone is defined as the area between the wall and d_{min}, which is the quadrotor's minimum stopping distance. When in this region, a collision with the wall is unavoidable. The width of the zone is a function of the quadrotor's approach velocity because the minimum stopping distance is dependent on the velocity. The minimum stopping distances for various initial approach velocities were found through simulation. The simulation was performed for two control modes, which will be described in Chapter 3. For the velocity control mode, a maximum velocity command was given in the opposite direction and the required stopping distances were recorded. For the angle control mode, a maximum angle command was given. A polynomial regression was done on the results of these simulations to create a predictive equation for the minimum stopping distance with a given initial velocity. Simulation results for the velocity control mode are shown in Figure 2-2. As seen in the figure, the stopping distance is a parabolic function of initial approach velocity. When the quadrotor is within the collision zone the force output is set to the maximum value to help minimize the collision speed.

2.1.2 Transition Zone

The transition zone provides a final warning before the quadrotor enters the collision zone, after which a collision is unavoidable. When the quadrotor enters this zone there is a jump in force output from the output in thr previous zone, and then the force increases asymptotically to the maximum value as the quadrotor penetrates further, until the force reaches the maximum value at the border of the collision zone. It has been observed that, without the transition zone, it is difficult for the user to discern when the collision zone is about to be reached.

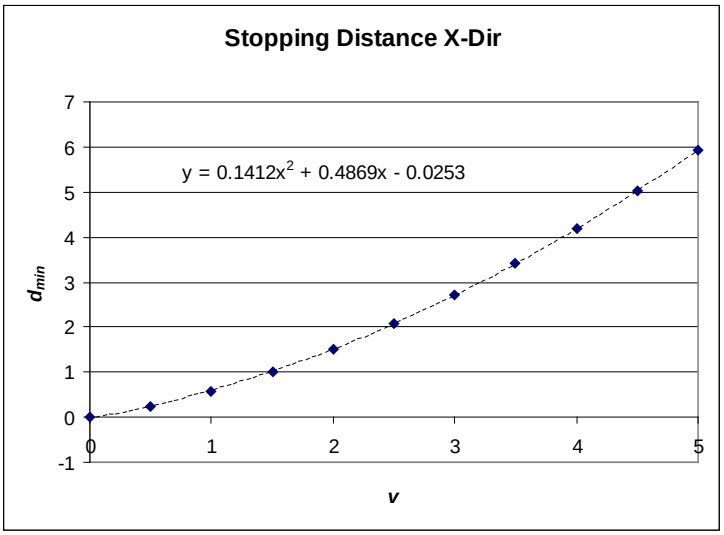

Figure 2-2: Minimum stopping distance (d_{min}) as a function of initial approach velocity (v) for the velocity control mode.

The distance from the obstacle that defines the beginning of this zone is d_{mid}, and is proportional to d_{min}. For example, d_{mid} could be set to $1.25d_{min}$, resulting in a transition zone that is 25% of the size of the collision zone. d_{mid} is made a function of d_{min}, rather than just being a constant value, to give the user the same amount of reaction time; if d_{mid} were a constant distance, the time taken to travel through the transition zone would decrease as the velocity increases.

2.1.3 Warning Zone

The warning zone is defined by two distances, $d_{desired}$ and d_{max}. The desired stopping distance is $d_{desired}$, which is a subjective comfortable stopping distance, and was determined through simulation with various initial approach velocities. In the simulation, a zero-velocity command was applied and the required stopping distances required were recorded. A linear regression was performed to determine the desired stopping distance for a given velocity. In the

case of the angle control mode, a command was given to put the quadrotor at half the maximum angle and a polynomial regression was performed on the results. Simulation results for the velocity control mode are shown in Figure 2-3.

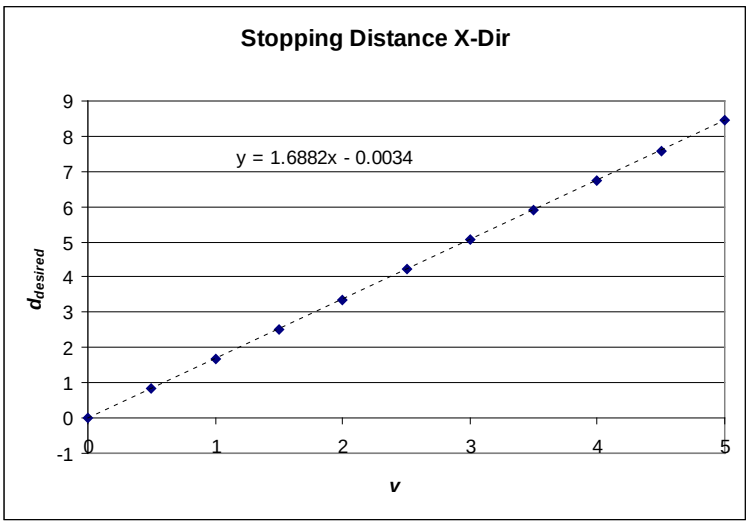

Figure 2-3: Predictive equation for desired stopping distance

The distance d_{max} is equal to $d_{desired}$ plus a constant value. This constant was added so that even when the quadrotor has little or no velocity, there will still be a buffer between it and an obstacle. All of the other distances are velocity dependent, so when the velocity is low they will have little effect. The purpose of this buffer is to prevent the quadrotor from slowly drifting into obstacles.

d_{max} defines the beginning of the warning zone. Before the warning zone, in the safe zone, no force is output to the user. In the warning zone, the forces increase linearly from 0% to 60% of the maximum force. When the quadrotor reaches d_{mid} the forces jump up to 80% and then increase up to 100% of the maximum force when d_{min} is reached. The force output with respect to velocity and distance is shown in Figure 2-4.

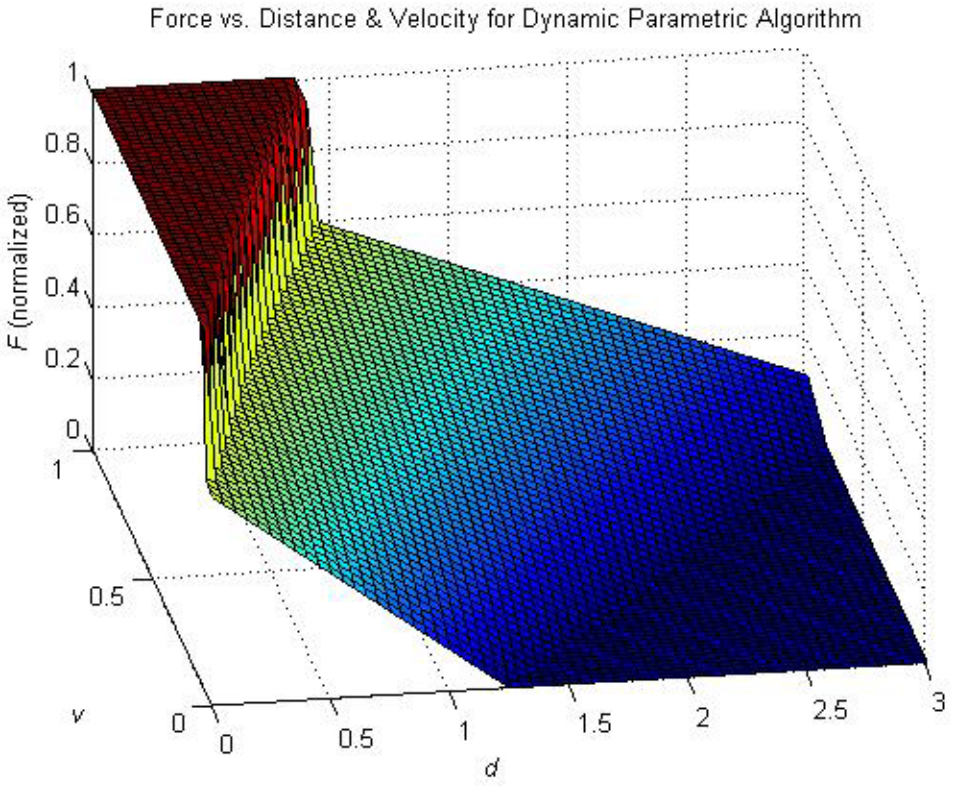

Figure 2-4: Force output as a function of velocity and distance for the Dynamic Parametric Field

The forces for the dynamic parametric algorithm are

$$F(d) = \begin{cases} d > d_{\max} & 0 \\ d_{mid} < d < d_{\max} & 0.6F_{\max}\left(1 - \dfrac{d - d_{mid}}{d_{\max} - d_{mid}}\right) \\ d_{\min} < d < d_{mid} & F_{\max}\left[0.2sqrt\left(1 - \left(\dfrac{d - d_{\min}}{d_{mid} - d_{\min}}\right)^2\right) + .8\right] \\ d < d_{\min} & F_{\max} \end{cases}, \qquad \textbf{(2-1)}$$

where *d* is the distance to the wall and *v* is the velocity, and

$$d_{max} = d_{desired} + x_{buffer} \qquad x_{buffer} = constant \tag{2-2}$$

$$d_{desired}(v) = 1.6882v - 0.0034 \tag{2-3}$$

$$d_{mid} = d_{\min} \cdot c \qquad 1 < c < 1.5 \tag{2-4}$$

$$d_{\min}(v) = 0.1414v^2 + 0.4869v - 0.0253. \tag{2-5}$$

2.2 Time to Impact

The time to impact (TTI) algorithm is based on work by others, such as Balas [16], who have used inverse time to collision to assist drivers in maintaining safe distances while following other cars. As the TTI decreases the haptic force increases. The TTI is calculated from the current velocity *v* and distance *d* to the obstacle as

$$\mathrm{TTI} = \frac{d}{v}. \tag{2-6}$$

The force is inversely proportional to the time to impact

$$F = \frac{k}{\mathrm{TTI}}, \tag{2-7}$$

where *k* appropriately scales the forces. The time to impact is a measure of the time remaining (if the velocity does not change) before collision will occur, which is why the forces are inversely proportional. As time decreases, the force increases. Because the inverse TTI approaches

infinity as the vehicle approaches an obstacle, the force level is saturated at the maximum force level. The output force with respect to the velocity and the distances is shown in Figure 2-5.

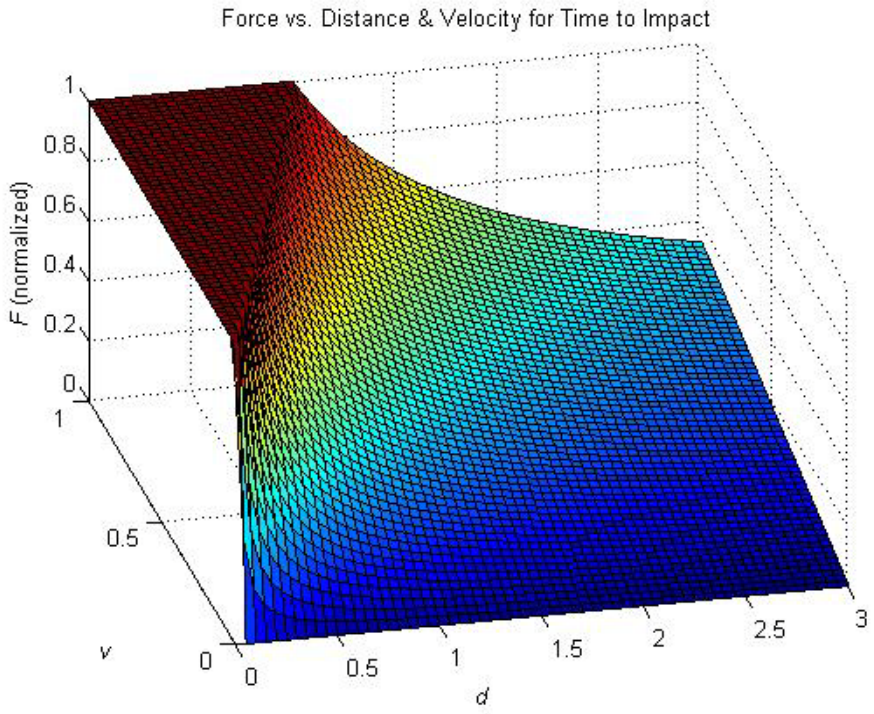

Figure 2-5: Force output as a function of velocity and distance for the time to impact algorithm

Two TTI algorithms were created. The first implements (2-6) and (2-7) directly. The second includes two modifications and the output force is shown in Figure 2-6. The first modification is that there is a TTI above which no forces are felt. This adds a safe zone in which the user will feel no forces. The other modification is that there is a minimum TTI at which the force jumps to the maximum allowable value F_{max}

$$F = \begin{cases} \text{TTI} > 3 & 0 \\ 3 > \text{TTI} > t_{\min} & \dfrac{k}{\text{TTI}} \\ \text{TTI} < t_{\min} & F_{\max} \end{cases}.$$

(2-8)

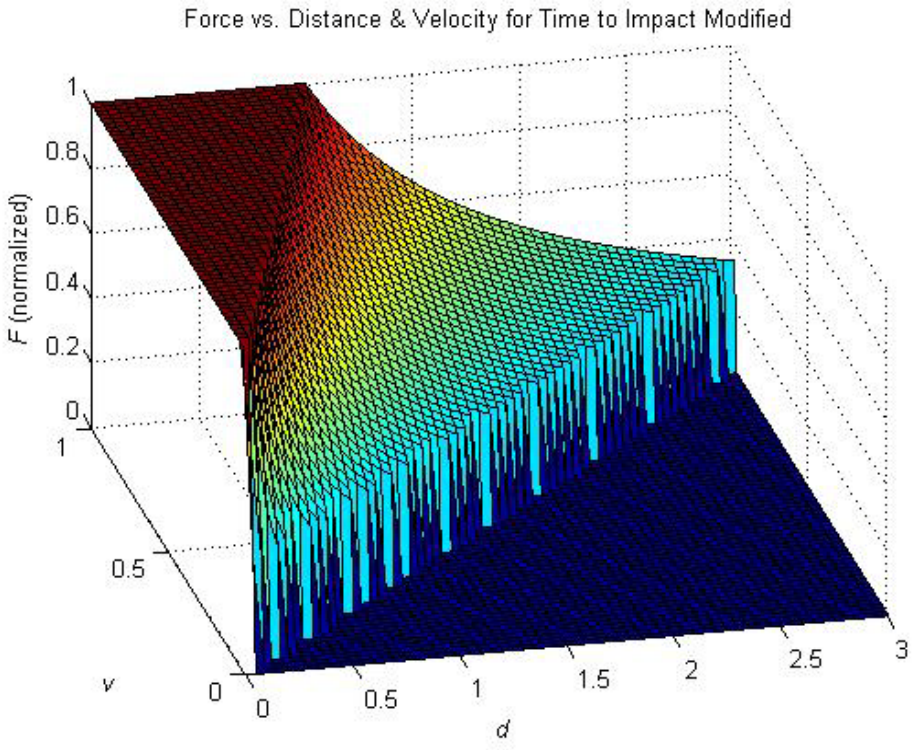

Figure 2-6: Force output as a function of velocity and distance for the modified time to impact algorithm

This minimum time t_{min} was found by using the d_{min} equation from the dynamic parametric algorithm and dividing both sides by the velocity as

(2-9) $$t_{\min}(v) = 0.1414v + 0.4869 - \frac{0.0253}{v}.$$

2.3 Spring Algorithm

The basic spring algorithm is based on the quadrotor's distance from obstacles, whereas the previous algorithms are functions of distance and velocity. In this algorithm, a virtual spring is connected between the quadrotor and the obstacle. As the quadrotor approaches the obstacle, the force increases linearly from zero at the outer limit of the spring region (which may be at the center of a hallway, for example) to the maximum force at the obstacle as shown in Figure 2-7. In many cases it may be desirable for the quadrotor to move away from the center of the hallway without a force being felt by the user. To make this possible, the spring force can be modified so that the spring region begins a distance d_{spring} away from the obstacle

$$F = F_{\max} \frac{d_{spring} - d}{d_{spring}}. \quad \textbf{(2-10)}$$

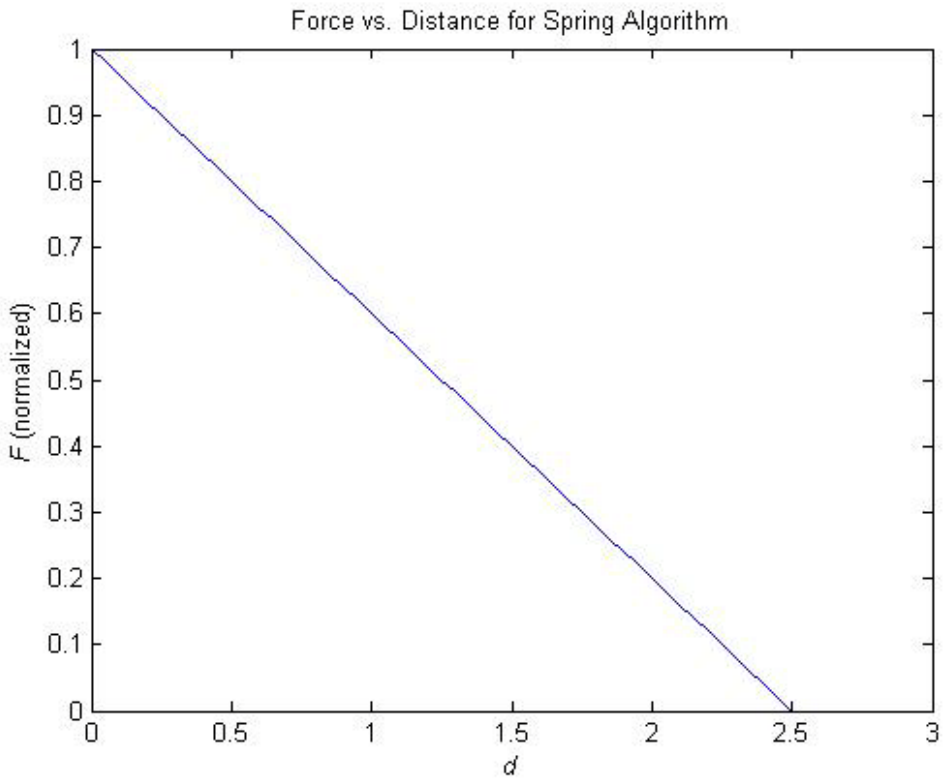

Figure 2-7: Force output as a function of distance for the spring algorithm

In a modification to the basic spring algorithm, the force multiplier is made to be dependent on the velocity of the quadrotor. The output force for this algorithm is shown in Figure 2-8. This does not change the dependence on d_{spring}, but only the magnitude of the forces felt within the spring area. The force multiplier increases as the speed of the quadrotor increases. The force goes from half of the maximum force to the full maximum force, by scaling the velocity with the maximum possible velocity

$$F = \left(F_{\max} \frac{v + v_{\max}}{2v_{\max}} \right) \frac{d_{spring} - d}{d_{spring}}. \tag{2-11}$$

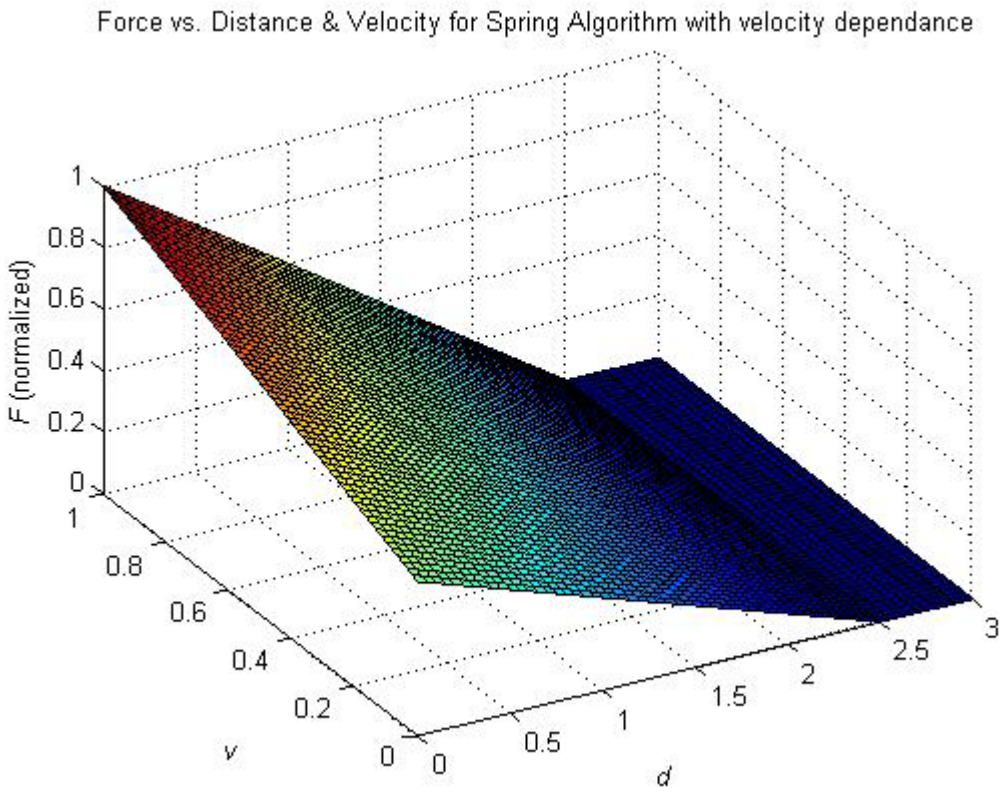

Figure 2-8: Force output as a function of velocity and distance for the spring algorithm with velocity dependence

2.4 Input Device Position Dependency

The force feedback algorithms described in the previous sections apply forces that help the pilot to avoid obstacles, but the forces were sometimes harmful. Because of the dynamics of the quadrotor, by the time the obstacle avoidance forces subside, the quadrotor accelerates away from the obstacle, resulting in an unintended velocity in this direction. In some cases, where obstacles are close together, the forces cause instabilities and collisions. To address this issue, another dependency was added to all of the algorithms: the position of the input device.

In each algorithm, the output is scaled between zero and one and then multiplied by the maximum force. With input device position dependency, instead of multiplying this output by

the maximum force, it is multiplied by 50 to get a desired input device position $p_{desired}$ because the input device area is limited to 50 mm in each direction, making the entire input device area 100x100x100 mm. For example, for the basic spring algorithm the desired input device position is given by

$$p_{desired} = 50\frac{d_{spring} - d}{d_{spring}}. \quad \textbf{(2-12)}$$

A force output of zero corresponds to the desired input device position at the zero position, and an output at the maximum corresponds to a desired input device position of 50. The actual position p of the input device is compared to the desired position and divided by 100, which is the maximum possible difference in positions. This scales the output between zero and one so it can be multiplied by the maximum force to get the force output to the haptic interface

$$F = F_{\max}\frac{\left(p - p_{desired}\right)}{100}. \quad \textbf{(2-13)}$$

This is analogous to having a spring connected to the input device with the zero position of the spring changing according to the output from the force feedback algorithm. It rewards the user for doing the correct action. If the user moves the input device in the direction necessary to avoid a collision, the force decreases. However, if the user does not move it in the correct direction the forces increase.

The force output as a function of the input device position and the desired input device position is shown in Figure 2-9. When the desired position is 50, meaning the algorithm wants the input device to be all the way forward, and the input device is at -50, meaning that it is all the way back, the output is the maximum force. If instead the input device were at 0 or 50, the output would be half the maximum force or 0, respectively. When the desired position is at 0

and the input device is at -50, the output would be half the maximum force, and if the input device were at 0, there would be no force output.

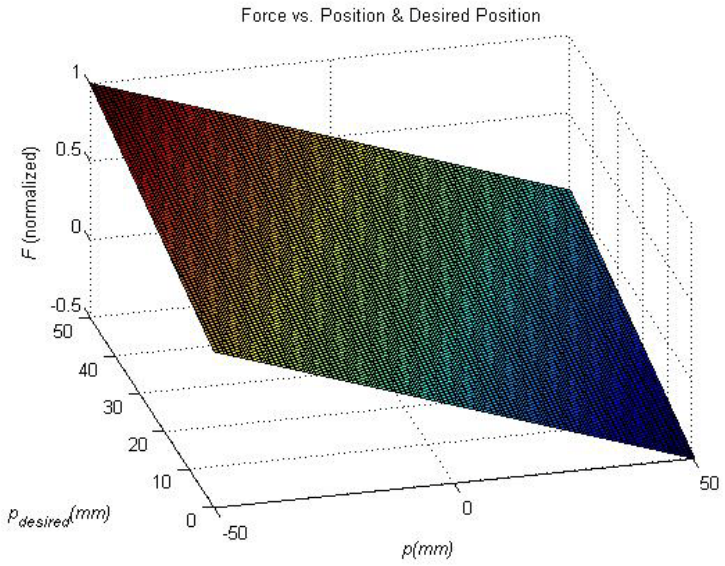

Figure 2-9: Force output as a function of position and desired position output from the force feedback algorithms

Figure 2-10 illustrates an example of how the input device position dependency works. In this figure, each box represents the X-Y plane of the input device workspace, the blue circle represents the current position of the input device, and the red diamond represents the desired position as dictated by the force feedback algorithms.

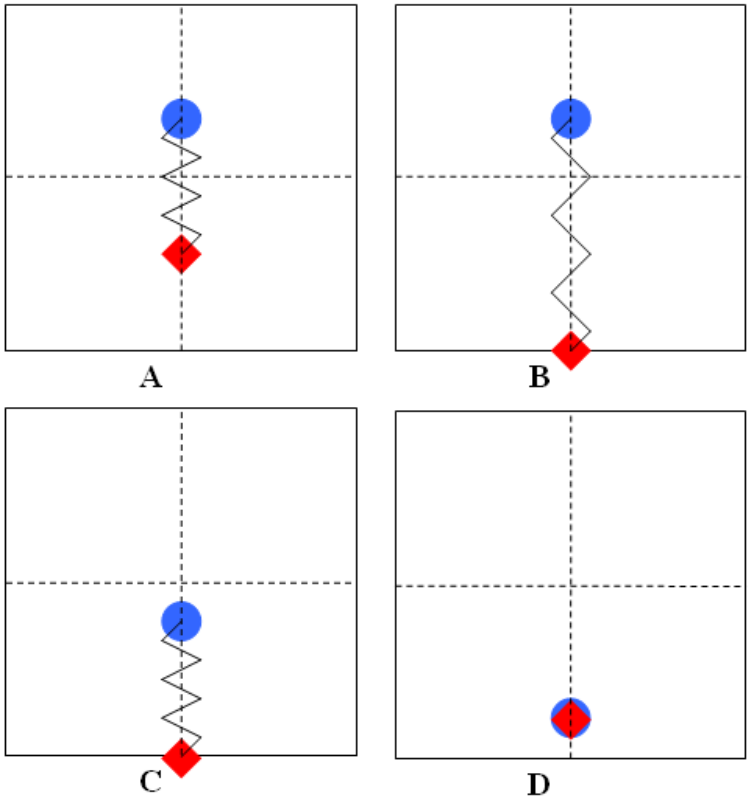

Figure 2-10: Example of the forces changing as a function of the position of the input device and the desired position output by the force feedback algorithms

In Figure 2-10 (A), the input device is commanding forward motion and the quadrotor is near an obstacle to the front, causing the feedback algorithm to give a desired position where the diamond is. The spring between the two represents the force pulling the input device to the desired position. In (B) the input device has not been moved and the quadrotor has gotten closer to the obstacle. The desired position has moved further back and the effective spring force pulling the input device has increased. Now the input device is pulled back and is closer to the desired position and the force is decreased as shown in (C). In (D) the position of the input device is the same as the desired position and there is no force pulling the input device in either direction.

2.5 Algorithm Summary

Five force feedback algorithms were developed to determine the forces output to the pilot through the force feedback device. The dynamic parametric field is based on the dynamics of the quadrotor and has different zones that determine how the forces are output. Two algorithms are based on the time remaining before the quadrotor comes into contact with an obstacle. The last two algorithms are based on the idea of a virtual spring between the wall and the quadrotor. Input device position dependency was added to all of the algorithms to prevent the force feedback from causing instabilities and collisions.

3 Simulation System

Force feedback algorithms for control of quadrotors were described in Chapter 2. In this chapter the hardware and software for simulating the force feedback algorithms and quadrotor dynamics are described. The simulation was built using Simulink, a graphical simulation environment within MATLAB. Simulink was used to model the quadrotor dynamics and the autopilot functions, and the SensAble PHANTOM was used to command the simulated quadrotor and provide force feedback to the user. An overview of the system is shown in Figure 3-1.

3.1 System Overview

The position values of the PHANTOM are read in through the Force function shown in Figure 3-1 and then passed to the Autopilot function. The autopilot interprets the positions as either desired velocities or desired angle states, depending on the application. The autopilot receives the states of the quadrotor from the Quadrotor Dynamics function to be used in the control loops and then calculates the voltages to be sent to the motors on the quadrotor. The Quadrotor Dynamics function turns these voltages into force values from each rotor and then calculates the states of the quadrotor at the next time step. These states are sent back to the force function which gives a graphical display of the quadrotor in a hallway as if looking through a camera on the quadrotor. The force function also uses the states of the quadrotor as well as the

positions of the phantom to calculate appropriate forces according to a force feedback algorithm. These forces are then sent to the PHANTOM to be felt by the hand of the user. The individual components will now be described.

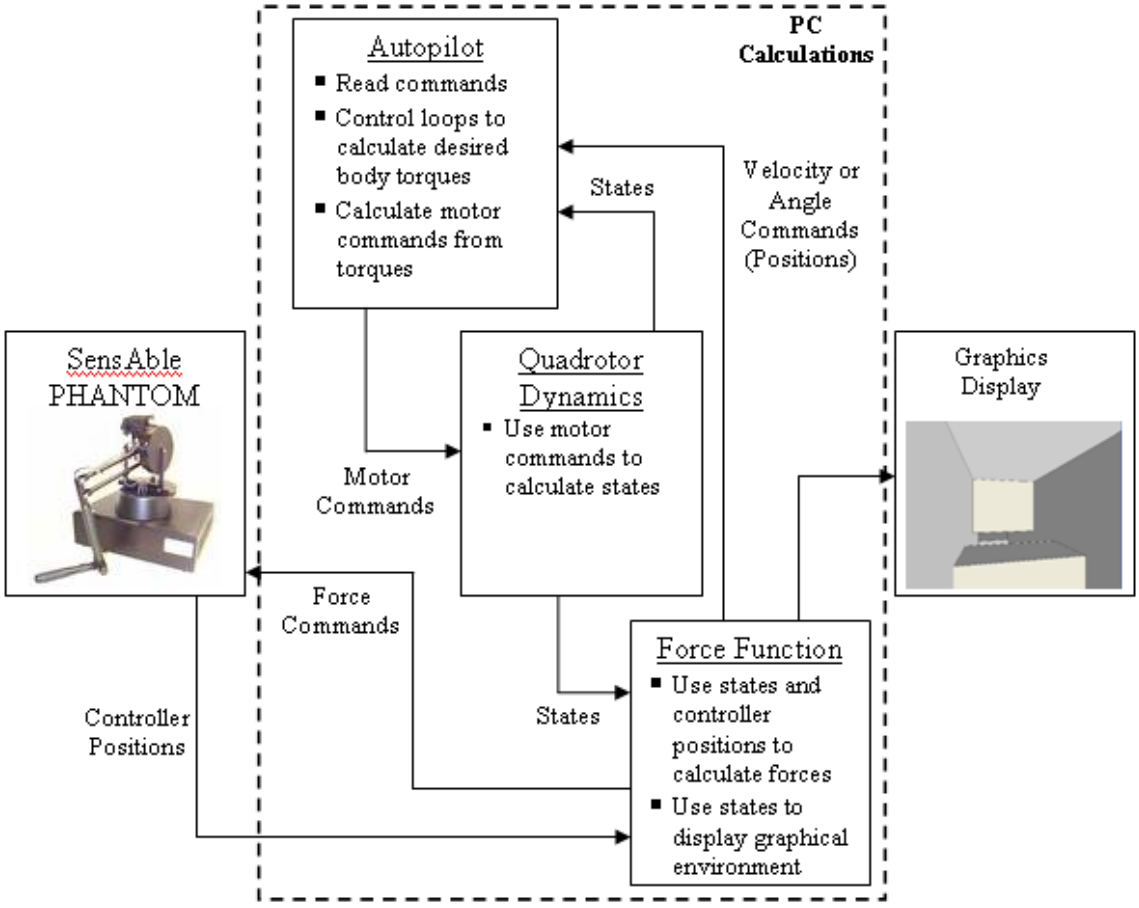

Figure 3-1: Overview of the main components and functions involved in the quadrotor simulation

3.2 Control Modes

Two types of simulation systems were developed. In *velocity control mode*, the forward and lateral displacements of the input device correspond to velocity commands in the body-fixed forward and lateral directions, respectively. In *angle control mode*, the displacement of the input device corresponds to pitch and roll angle commands. In both modes a vertical input displacement corresponds to a vertical velocity command and a rotational displacement of the input device corresponds to a yaw angle rate command. In velocity control mode, velocity commands were chosen because they make the quadrotor more intuitive to control compared to commands. When the input device is moved forward the quadrotor moves forward, when the input is brought to the center of the workspace the quadrotor comes to a stop, and when the input is move backwards the quadrotor moves backwards. The problem with velocity control mode is that it requires accurate velocity estimates to be used in the control loop. An onboard system to determine velocities accurately in indoor environments has not yet been developed. Also, the quadrotor used for flight testing, as described in Chapter 6, does not have the capability to do velocity control. The quadrotor used in this study accepts either angle or angular rate commands. A simulation was also made to take angle commands to simulate the control system that is used for flight testing. Angle commands were chosen instead of angle rate commands because commanding angles was found to be more intuitive and easy to control.

3.3 SensAble PHANTOM

The SensAble PHANTOM (Figure 3-2) is a commercial haptic device with 6 degrees of freedom (DOF) sensing capabilities: *x, y,* and *z* positions, and roll, pitch, and yaw angles. The

PHANTOM also has three force degrees of freedom, allowing it to generate forces in the *x*, *y*, and *z* directions. The maximum force output in each direction is 8.5 N [17]

The PHANTOM provides an intuitive interface for commanding a quadrotor, since the six sensing degrees of freedom of the PHANTOM may be mapped onto the six motion degrees-of-freedom of the quadrotor. For a quadrotor, the pitch and forward translation degrees of freedom are coupled. Similarly, roll and lateral motion are coupled. As a result of these coupled degrees of freedom, only four independent command variables are required to control the quadrotor: forward translation or pitch, lateral translation or roll, vertical translation, and yaw angle.

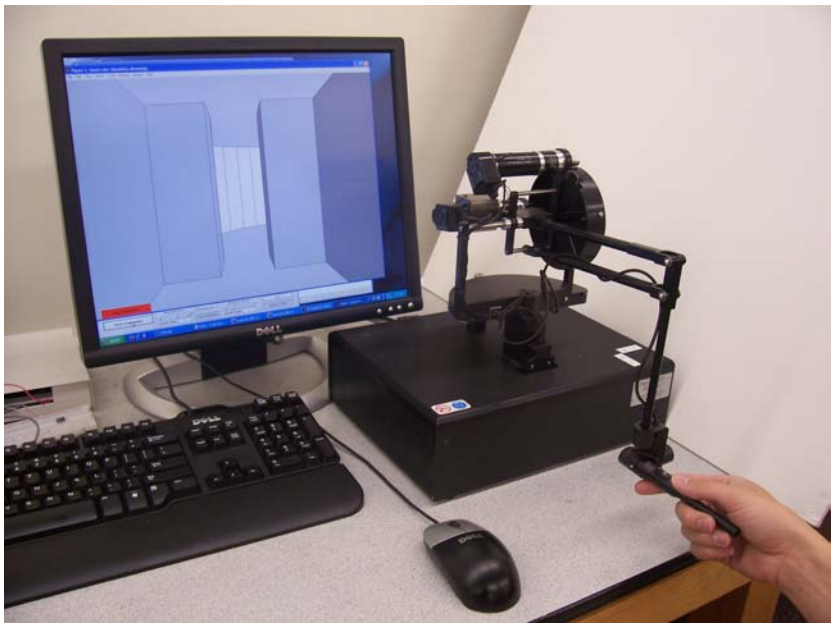

Figure 3-2: The SensAble PHANTOM Premium 1.5 used to control the quadrotor

For this thesis, displacements of the PHANTOM handle in the *x*, *y*, and *z* directions are mapped into velocity commands in the forward, lateral, and vertical quadrotor directions,

respectively, for velocity control mode. They are mapped in to the pitch angle, roll, angle, and velocity in the vertical direction, respectively, for angle control mode. The horizontal rotational angle of the handle is mapped into desired quadrotor yaw rate.

The PHANTOM connects to the computer through the parallel port and can communicate with the computer through a C++ library created by SensAble. It is able to communicate with Matlab through a DLL library that was created for this thesis. Five Matlab functions were created: an initialization function, a shutdown function, a function to read in the *x*, *y*, and *z* positions, a function to read in the three gimbal encoder angles, and a function to apply forces in the *x*, *y*, and *z* directions. These five functions give Matlab the functionality necessary to be the platform for this simulation.

3.4 Simulink Model

The commands generated by the user using the PHANTOM are read into a Simulink model that simulates the quadrotor dynamics and autopilot, calculates feedback forces based on the algorithms described previously, and generates the graphics for a simulated indoor environment. The highest level of the Simulink model is shown in Figure 3-3. The PHANTOM commands are read in the Force/Plotting block. These commands are passed to the Autopilot function where commands to the individual quadrotor motors are generated. The Dynamics block simulates the complex quadrotor equations of motion in response to the motor commands. The states of the quadrotor are sent to the Force/Plotting block, where feedback forces are computed, based on the force feedback algorithms described in Chapter 2, and sent to the PHANTOM. The loop rate of the Simulink model was found to be about 30 Hz. The Force/Plotting, Autopilot, and Dynamics blocks will be described in more detail below. The

other blocks shown in Figure 3-3 are for handling input parameters, data collection, and timing functions, and will not be described in further detail.

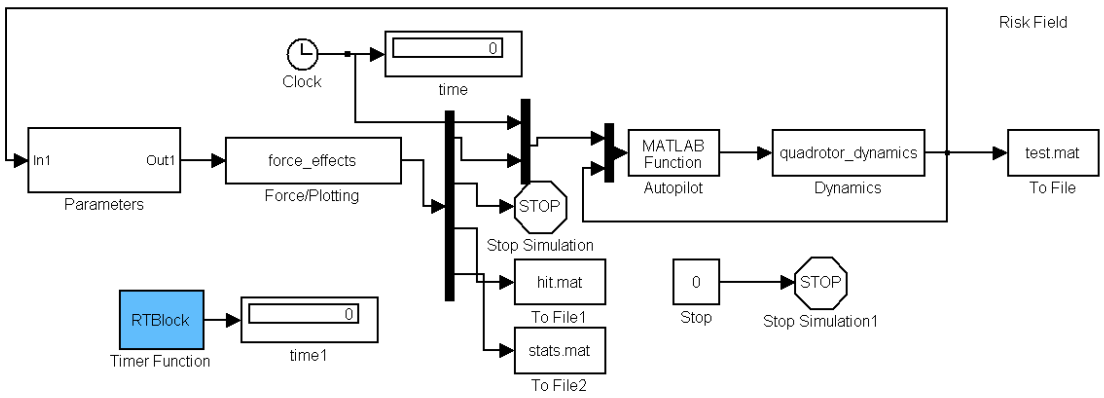

Figure 3-3: The Simulink block diagram used to simulate the quadrotor with haptic feedback

3.4.1 Autopilot Function

The autopilot function was developed by Dr. Randal Beard, a professor in the Electrical Engineering Department at Brigham Young University, but modified by changing some control loops and gains for this application. It receives the velocity or angle rate commands and quadrotor states, and generates quadrotor motor commands. In the velocity command mode, the actually velocity is compared to the desired velocity and then, using a PD control loop, it calculates a desired roll or pitch angle in the case of *x* and *y* directions. In the case of the *z* direction, the loop outputs a thrust force. The desired roll and pitch angles are compared to the actual angles and then sent through another PD control loop that calculates the torques about the body *x* and *y* axes. For the yaw rate, the desired and actual rates are compared and sent through one PD control loop to calculate a *z* axis torque. The thrust and torques are then multiplied by known parameters of the quadrotor, such as mass and moments of inertia, to determine motor

commands to be sent to the quadrotor. In angle command mode, the first control loops, in which the velocities in he *x* and *y* directions are compared, are not used.

3.4.2 Quadrotor Dynamics Function

The motor commands are received by the quadrotor dynamics function and used to calculate the torques and forces that act on the quadrotor body. These forces and torques are used as inputs to the quadrotor equations of motion given by

$$\textbf{(3-1)} \qquad \begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{pmatrix} = \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi - c\phi c\psi & c\phi s\theta s\psi + s\phi c\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

$$\textbf{(3-2)} \qquad \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}$$

$$\textbf{(3-3)} \qquad \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\theta\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

$$\textbf{(3-4)} \qquad \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \dfrac{J_y - J_z}{J_x} qr \\ \dfrac{J_z - J_x}{J_y} pr \\ \dfrac{J_x - J_y}{J_z} pq \end{pmatrix} + \begin{pmatrix} \dfrac{1}{J_x}\tau_\phi \\ \dfrac{1}{J_y}\tau_\theta \\ \dfrac{1}{J_z}\tau_\psi \end{pmatrix},$$

where

p_n = the inertial north position

p_e = the inertial east position

h = the altitude of the quadrotor

u = the body frame forward velocity

v = the body frame lateral velocity

w = the body frame vertical velocity

φ = the roll angle measured in the vehicle 2 frame

θ = the pitch angle measure in the vehicle 1 frame

ψ = the yaw angle measure in the vehicle frame

p = the roll rate measured in the body frame

q = the pitch rate measured in the body frame

r = the yaw rate measured in the body frame.

These equations were derived by Dr. Randal Beard [18] and are used with Simulink's built in ODE solver, which calculates updated state variables at each time step.

3.4.3 Force/Plotting Function

In this block, the feedback forces are calculated as a function of the quadrotor states and the quadrotor's relationship to the simulated indoor environment. The quadrotor states are also used to generate the appropriate camera view in the simulated environment.

To determine the forces to feed back to the user via the PHANTOM, the x, y, and z positions of the quadrotor are compared with the known location of the nearest obstacles in the world frame. This means that the distances are calculated in the world frame, whereas the distances on the real quadrotor will be found in the body frame. It can be shown that the force output will be the same in either case. An example is given in Figure 3-4 using the time to impact algorithm. Depending on the force feedback algorithm (see Chapter 2), the distances are

either used with only the position of the PHANTOM or both in combination with the Cartesian velocities to calculate the forces that will be applied to the hand of the user.

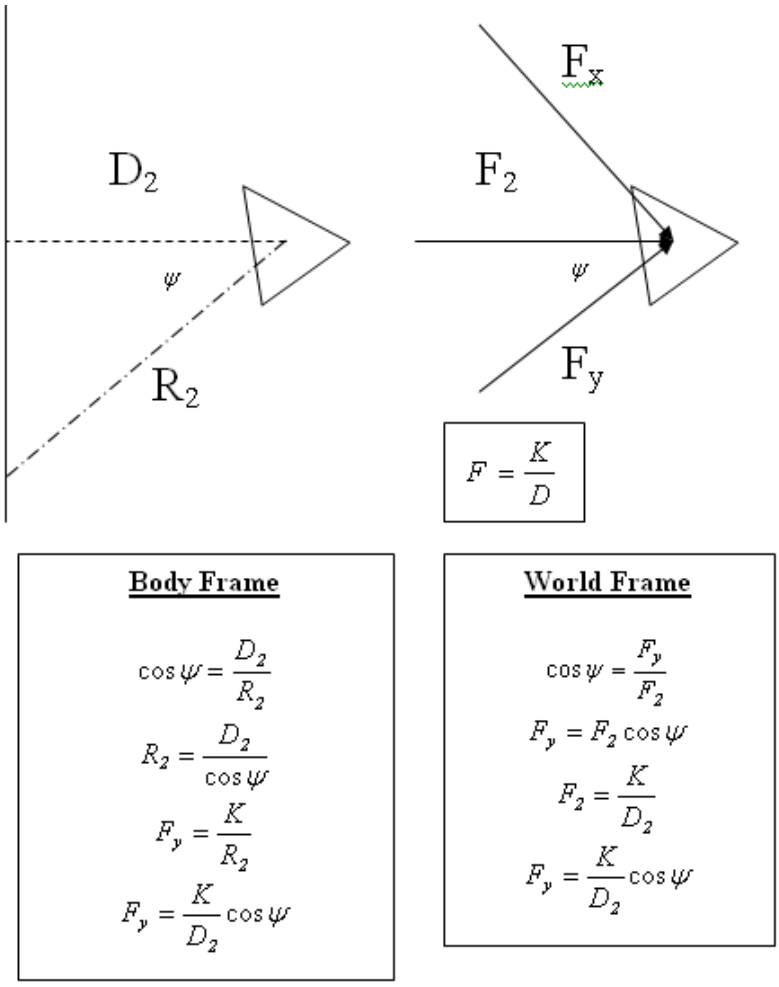

Figure 3-4: Example of how forces are the same whether distances are found in the world frame or the body frame

The force function is also used to make the quadrotor feel like a traditional joystick. First, a spring force is used to help bring the PHANTOM back to its zero position. This is done by exerting a force proportional and opposite to the displacement of the PHANTOM. This centering force is included because it is difficult for a user to know where the zero position is

without looking at the controller. It is important that this force is smaller than the collision avoidance forces, so as not overpower them.

Another force that is added is a controller area limiting force. The workspace of the PHANTOM is very large and not symmetrical. This force limits the workspace of the phantom to about a six inch cube. It is implemented by simulating a very stiff spring when the displacement of the PHANTOM exceeds a certain value. This keeps the controller within the range specified and its feel is significantly different from the collision avoidance forces so the two are not confused.

3.5 The Indoor Environment

The purpose of this thesis is to explore force feedback for piloting quadrotors in indoor environments. A simulated indoor course was created to enable testing and refinement of the force feedback algorithms presented previously. The simulation course consists of a hallway with two turns and various other obstacles, as shown in Figure 3-5.

The width and height of the hallway is 10 times the width of the quadrotor. The quadrotor begins the course in the middle of the hallway where it is pictured in Figure 3-5. In Zone 1, outlined by the dashed red lines, there is wind that pushes to the left. This is to see if the force feedback algorithms are robust against wind and also to move the quadrotor out of the middle of the hallway to make the doorway (Obstacle 2) more difficult to pass through. Obstacle 3 is a curved wall to test the effects of the algorithms against non-straight surfaces. The quadrotor must then traverse over Obstacle 4 and under Obstacle 5. Obstacle 6 is an area of almost total darkness to test the effect of having very little visual information. Obstacle 7 is a

pillar placed in the middle of the hallway. After the pillar, the user goes toward the end wall and the simulation stops automatically when within a certain distance of this wall.

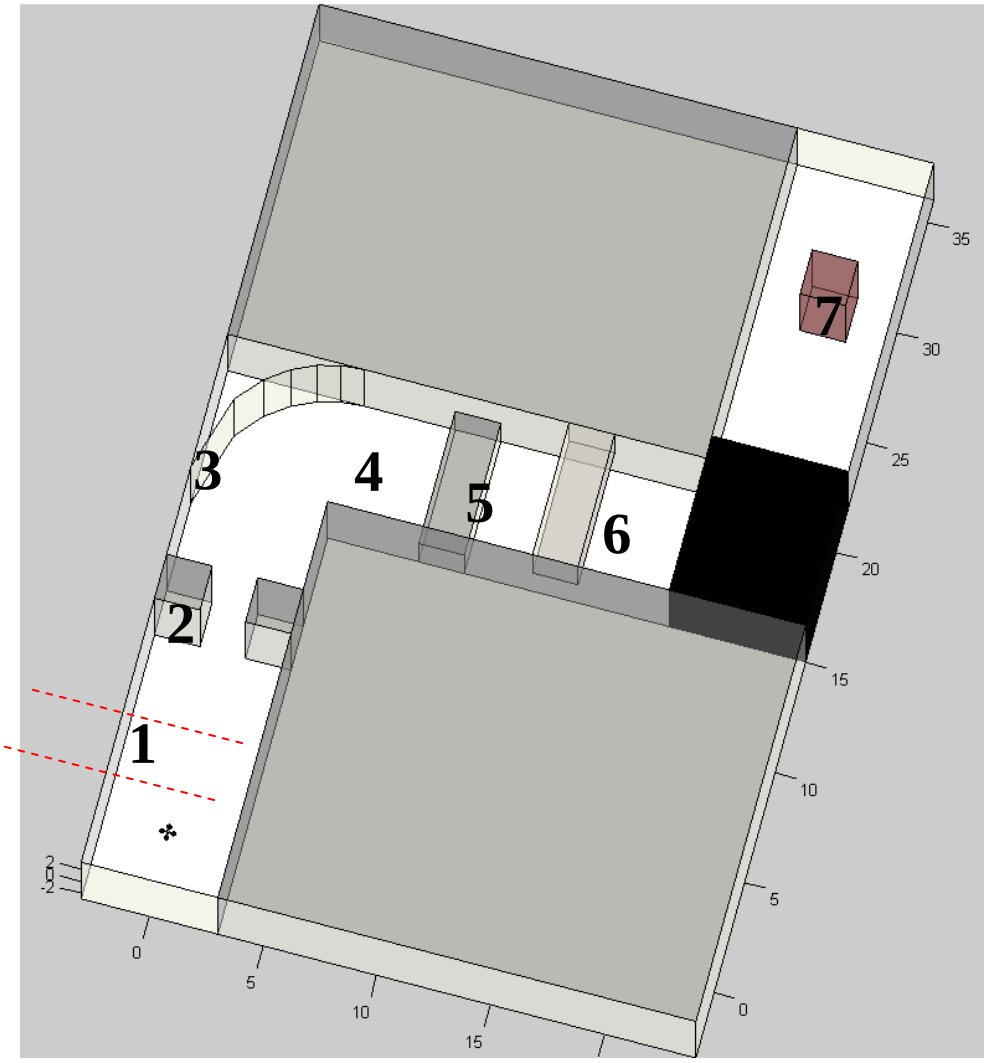

Figure 3-5: Overhead view of the hallway course

During the simulation, the hallway and obstacles are plotted using the patch function in Matlab which creates 2 dimensional patches in the 3D world. The view of the camera is placed at the *x*, *y*, *z* location of the quadrotor and oriented using the rotational states of the quadrotor. At each time step, the camera is moved to the current *x*, *y*, *z* coordinate and rotation, which creates

the impression of a continuously running camera. Three sample views created in this way are shown in Figures 3-6 through 3-8.

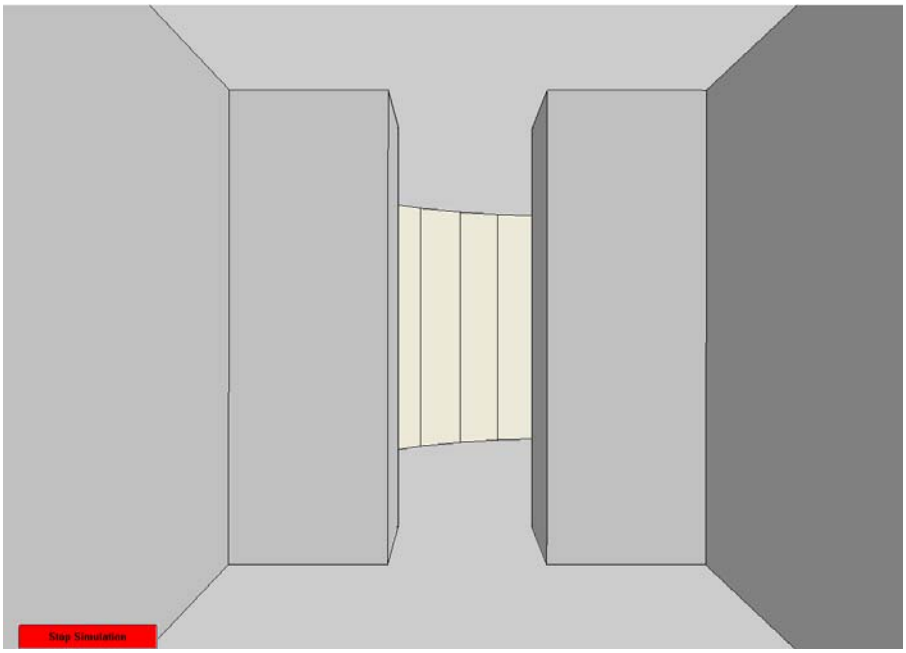

Figure 3-6: Camera view at the course beginning

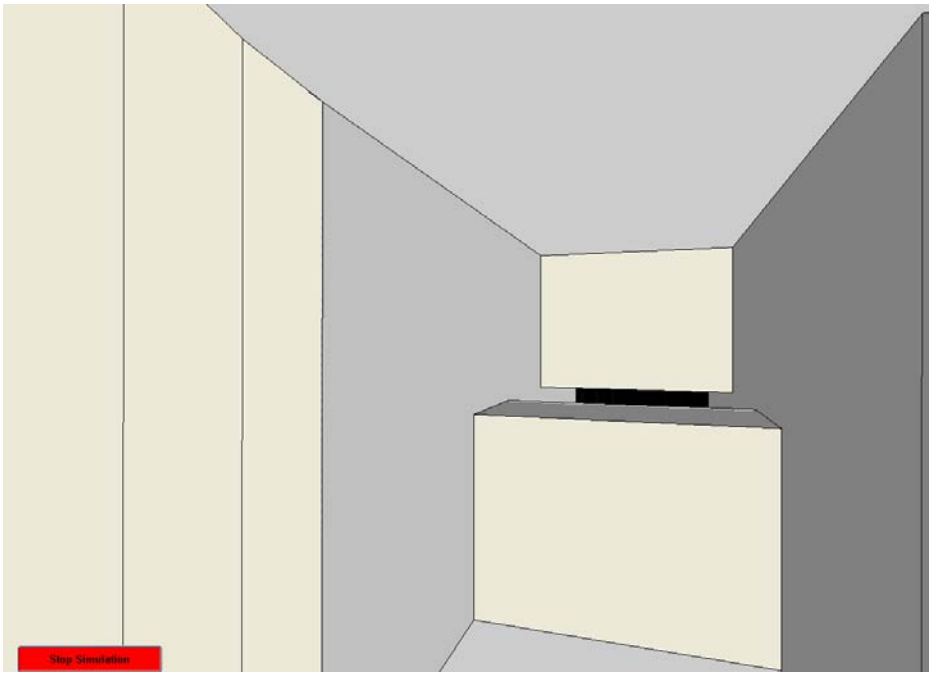

Figure 3-7: Camera view from the quadrotor after making the first turn

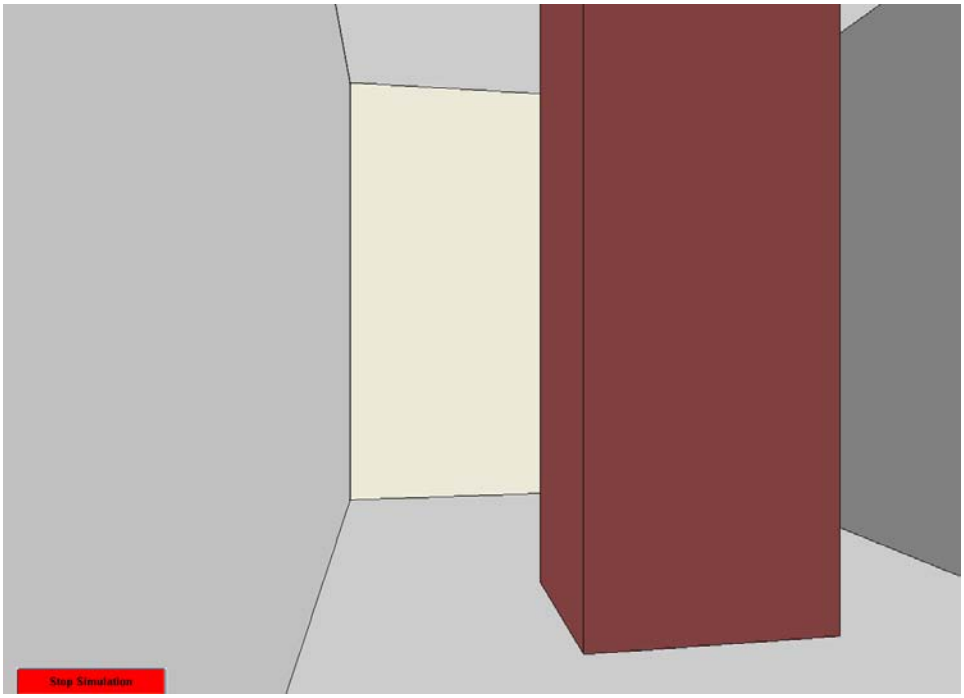

Figure 3-8: Camera view from the quadrotor looking at the pillar after the dark area

3.6 Other Simulation Features

The simulation is controlled using a graphical user interface shown in Figure 3-9. On the left in the Force Algorithm box are buttons that allow the user to choose which algorithm to use for the force feedback. The buttons in the Simulation Mode box switch between training and testing mode. In training mode, all parameters can be chosen freely and in testing mode the parameters are pre-programmed and the user is not shown what the parameters are. This is for statistical testing to prevent bias in the experiment. The other parameters that can be changed (from left to right along the bottom) are the force level, whether there are forces or not, the controller sensitivity (identified as Maximum Velocity), overhead view or camera view, and black area or no black area. The auto centering button was for a feature that is no longer used.

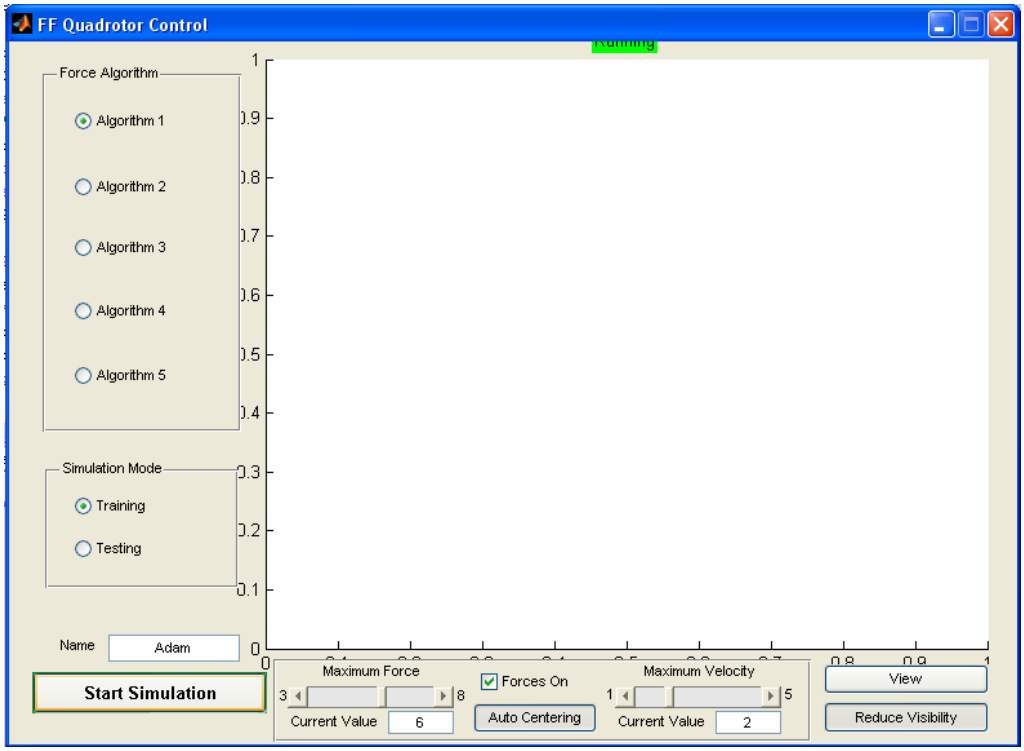

Figure 3-9: GUI used to control the quadrotor simulation

Above the start button in the bottom left corner is a box for the name of the current user. When the user enters a name, a file is created to store simulation data about that user's runs. During the simulation, data are stored in temporary files. These data include all states, force information, where hits occur, etc. After the simulation ends, a function is called that creates and saves a plot of the quadrotor path and calculates and saves statistical data about the runs. These statistics will be discussed in Chapter 4. On the plot of the quadrotor path, the collisions are highlighted by large red dots, from which the number of times the user hit can be counted. An example of this is shown in Figure 3-10.

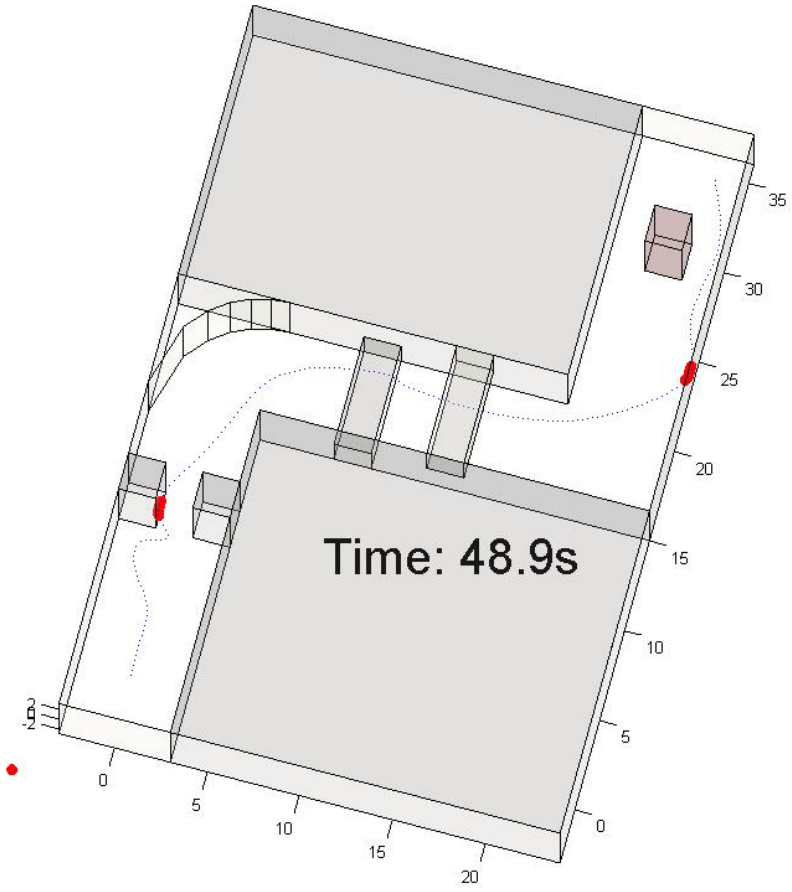

Figure 3-10: An example output path for a user showing the hits in red and the time taken to complete the course

3.7 Summary

A simulation was created using Simulink to simulate a quadrotor flying through an indoor environment. It is interfaced with a human operator using the SensAble PHANTOM commercial haptic device. The simulation uses an autopilot function that sends commands to a dynamics function that uses the equations of motion of a quadrotor to calculate the states at each time-step of the simulation. A force function uses the states of the quadrotor along with the positions of the PHANTOM to calculate the feedback forces and send them to the PHANTOM.

An indoor environment is created with several obstacles. A GUI allows the user to change parameters and begin the simulation, and information about each run is recorded when the simulation ends.

4 Human Subject Experiment 1 – Velocity Control

An experiment involving human subjects was conducted to test the force feedback algorithms using velocity control of a simulated quadrotor. The purpose of the experiment was to determine if force feedback helps pilots in the remote operation of a quadrotor and, if so, which algorithm is the most beneficial. Another purpose was to find if the level of force has an effect on the operation and find out which force level is most beneficial.

4.1 Factors and Factor Levels

Two factors were chosen for testing: force feedback algorithm and force level. There are many other factors that may affect the control of quadrotors in indoor environments, but because the purpose of this thesis is to explore the effects of force feedback only, no other factors were considered.

For the force feedback algorithm factor, five levels were tested. The five levels are the five feedback algorithms discussed in Chapter 2: (1) dynamic parametric field, (2) time to impact, (3) time to impact with a no-force zone and minimum time, (4) spring algorithm, and (5) spring algorithm with a velocity-dependent time constant. The second factor, force level, is the tunable maximum force parameter in the algorithms listed above. The PHANTOM interface has a constant force output range of 0 to 8.5 N, so three levels within that range were tested: 3, 5, and 7 N. Testing at these three force levels adequately spans the force range.

To determine if piloting a quadrotor with force feedback is superior to piloting without force feedback, runs with no force feedback were added to the experiment. This is an additional level of the feedback algorithm factor, but because it cannot be tested at different force levels it is treated as a separate factor.

4.2 Experiment Design

4.2.1 Experimental Setup

Typical statistical Design of Experiments (DOE) methods of having a number of factors each with a number of levels could not be employed because the experimental runs without force feedback do not fit in as a level of either factor. The experiments were designed in consultation with the BYU Center for Collaborative Research and Statistical Consulting. The experiment was set up to have fifteen users that complete six runs each. Five of the runs were a random subset of the five algorithms/three force level combinations. Another run that has no force feedback was placed randomly within that subset. Between all fifteen users, each combination was tested 5 times and the no-force case was tested 15 times. The resulting experiment design is an incomplete block partially confounded factorial. The reason it is incomplete is because each subject's block of runs does not span all possible combinations. The experiment is partially confounded because one run (no force feedback) is tested more than the others. An experiment that is not confounded would have all combinations tested an equal number of times.

4.2.2 Testing Procedure

All of the experiments were performed under Brigham Young University IRB protocol 01-0170 and all volunteers were given adequate time to read and sign a consent form before

participating in the experiment. The volunteer was then brought to the simulator station and was given instructions. First, the user was told what a quadrotor is and was made aware of its omni-directional capabilities. The force feedback capabilities of the PHANTOM were then explained and the user was shown how to guide the quadrotor and given an example by the test administrator. The objective was explained as completing the simulated indoor course without any collisions and as quickly as possible. Both goals were given equal weighting for this experiment. The user was told that the force feedback algorithms would change for each run, but they would not be told which algorithm it was, and they would not be able to see any results until all of the runs were completed.

All subjects had never used the simulator, nor seen it used prior to the experiment. Each was given two practice runs with the objective of familiarizing themselves with the simulator. After the practice runs, they were asked to complete six consecutive experimental runs. After each run the user would fill out a form to measure their workload. When all six runs were completed, they would fill out a questionnaire to asses the overall experience. All users were treated equally and precautions were made to remove all possible bias from the experiment, such as randomizing the runs and giving equal treatment to the volunteers.

4.3 Measures

Several measures, or outcomes, were chosen to assess the effectiveness of the force feedback algorithms. They are divided into two categories, direct and indirect measures. The direct measures indicate whether the force feedback prevented costly collisions and if it helped or hindered the pilot in efficiently completing the tasks. The indirect measures do not tell how

well a force feedback algorithm performed, but are possible indicators as to why an algorithm performed better or worse. The direct measures are outlined as follows:

- Number of Hits – This measures how many times the quadrotor hits an obstacle. Because it is possible to scrape along obstacles, each continuous length of hit is recorded as one hit.
- Length of Hits – This measures the length of each hit using the Euclidean distance between hit points at each time step in the simulation and adds them together.
- Workload – The overall workload of user for each run, measured by the user with the NASA Task Load Index (TLX) developed by Hart and Staveland. [15]
- Time – The total time to complete the course.

The indirect measures are:

- Average Velocity – The average overall velocity of the quadrotor throughout the entire run.
- Standard Deviation of Velocity – The standard deviation of the velocity of the quadrotor throughout the run.
- Path Length – The entire distance travelled by the quadrotor for the run.

The following measures were recorded for each direction, *x*, *y*, and *z*, independently, as well as the Euclidean average of the three directions.

- Average Force – The average force (in Newtons) that the force feedback algorithm output to the haptic device during the entire run. Forces are output and recorded when the quadrotor hits an obstacle, even in the case of no force feedback.
- Standard Deviation of Force – The standard deviation of the force that the force feedback algorithm output to the haptic device.

- Input Device Command Average – The average distance (in mm) of the input device from its zero position.
- Input Device Command Standard Deviation – The standard deviation of the distance of the input device from its zero position.

4.4 Results and Analysis

The analysis of this experiment was done in two parts due to the complexity of the design. First, the sixteen possible runs were analyzed as separate treatments. Second, the no-force runs were ignored and the experiment was analyzed as if it were a 3x5 factorial, meaning that there is one factor with three levels and one factor with five levels. Both were analyzed using a mixed-model analysis of variance (ANOVA), treating the block as a random variable to take into account differences between users.

An ANOVA uses the variance of an outcome, or measure, to determine statistical differences between the means of that outcome of the different factor levels. It does this by using the variance of the measure to create normal probability distributions around the mean of each factor level. The farther away one level's mean is from another, the less probable it is that the means are equal and the variation is due to noise within the data. The probability of the means being equal is determined by the variance of the data. The output of the ANOVA is a probability that the two means are equal. If the probability falls below .05 than it is considered statistically significant that there is a difference in the means.

Using the mixed-model analysis allows the differences between users, such as one user performing much better or worse than another, to be accounted for and equalized when doing the ANOVA. This is because it allows analysis of fixed-effects, such as the changes in the factors,

along with the analysis of random effects, such as different people performing the experiment. By treating the different users as a random effect, the variance of each user and the variance between users can be accounted for to reduce the variance used in the ANOVA calculations, thus giving more power to discern differences between means. This equalizing is done because the experiment is seeking to find the differences between the factor levels, regardless of the user performing the experiment.

4.4.1 16 Treatment Analysis

To determine if any of the combinations were different from having no force feedback, a mixed-model ANOVA was done while treating each of the fifteen combinations and the no force runs as different factors, or treatments, instead of levels of two factors. If there was greater than a 95% confidence level that there was a difference for a measure, then each of the fifteen combinations were compared with the no force runs using the Dunnett simultaneous test [19]. This test uses one control variable, no-force, and compares it to all of the other treatments to determine which combinations were different and whether they are statistically higher or lower than no-force. It takes each treatment and calculates a test statistic that it compares to a critical value that is taken from a Dunnett table to determine the significance. The critical value depends on the number of samples in the treatment and the number of treatments being compared.

The measures that were significantly different (95% confidence level) were hits, hit length, workload, completion time, average velocity, standard deviation of the *z* command, and all of the force measures. The significance of the force measures is trivial since they are being compared to having no force, so those results were not analyzed further. The other results were compared using the Dunnett tests. For hits and hit length, it was found that only the time to impact algorithm at the highest force level was significantly different than having no force

feedback and that it reduced both. Only the spring algorithm with the velocity dependence at force level 7 N significantly increased the workload compared to having no force feedback. The spring algorithm at a force level of 5 N was the only algorithm to significantly increase completion time.

The results of the average velocity comparison showed that the time to impact algorithm at the 5 and 7 N force levels and the spring algorithm with the velocity-dependence at the 5 N level decreased the average velocity. It is possible that the lower average velocity of the spring algorithm is the reason that it had a higher time to completion. However, it is inconclusive because the time to impact algorithms did not show significantly higher completion times, despite the lower average velocity. The parametric algorithm at the highest force level had a lower standard deviation in the *z* command, from which no conclusions are apparent.

4.4.2 3 x 5 Analysis

For this analysis, the no-force runs were ignored and the data was analyzed as if the experiment were a 3x5 full factorial. Again, the ANOVA was done for each measure. If significance was found, an effects plot was created for the main effect or interaction to determine where the significance was. The outcomes that were found to be significant were hit length, workload, average and standard deviation in the *z* command, and all of the force measures.

The significance in the hit length and number of hits was found to be in the force level. From the main effects plot shown in Figure 4-1 it can be seen that increasing the force level from 3 to 5 N significantly decreased the hit length. The dashed line in each figure represents the overall data mean. For workload, the significance was in the algorithm and the interaction. These are shown in Figures 4-2 and 4-3. The main effects plot shows that the spring algorithms with the velocity dependence resulted in a higher workload than all of the other algorithms. The

interaction plot shows that at the lowest force level the workload was similar to that of the other algorithms, but at the higher force levels the workload significantly increased.

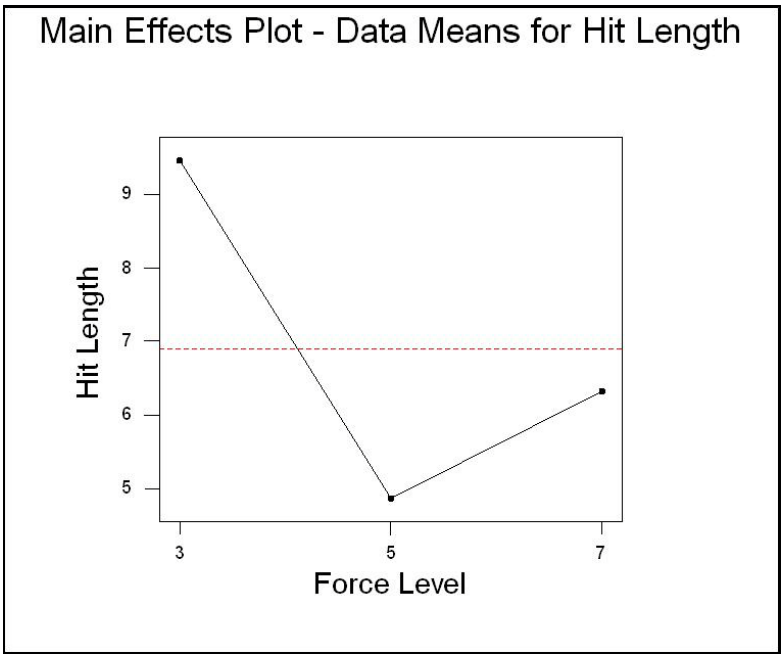

Figure 4-1: Main effects plot of hit length vs. force level for the 3 x 5 analysis of experiment 1

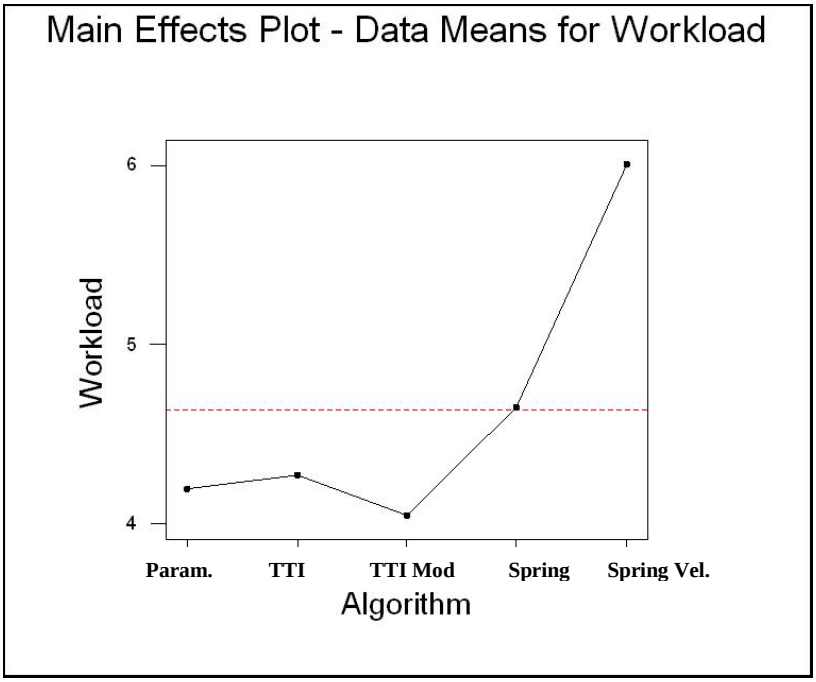

Figure 4-2: Main effects plot of workload vs. algorithm for the 3 x 5 analysis of expereriment 1

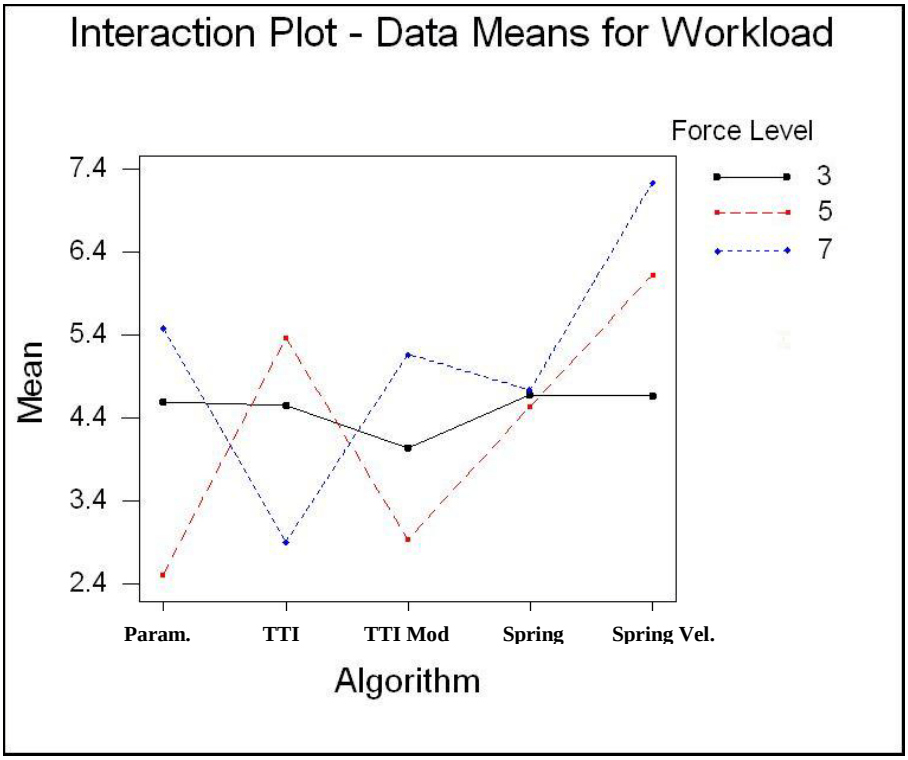

Figure 4-3: Interaction plot for workload, force, and algorithm

The force level had a significant effect on both the average and the standard deviation of the *z* command. This is shown in Figures 4-4 and 4-5. These plots indicate that the average and standard deviation of the *z* command decreased between 3 and 5 N but did not have a significant change between 5 and 7 N. This is most likely due to the centering force of the controller. The higher forces made the centering force higher, which reduced movement in the *z* direction. It is possible that because the higher centering forces caused a significant effect, they may also have some influence on how well the force feedback algorithms performed. This is something that could be investigated further in the future.

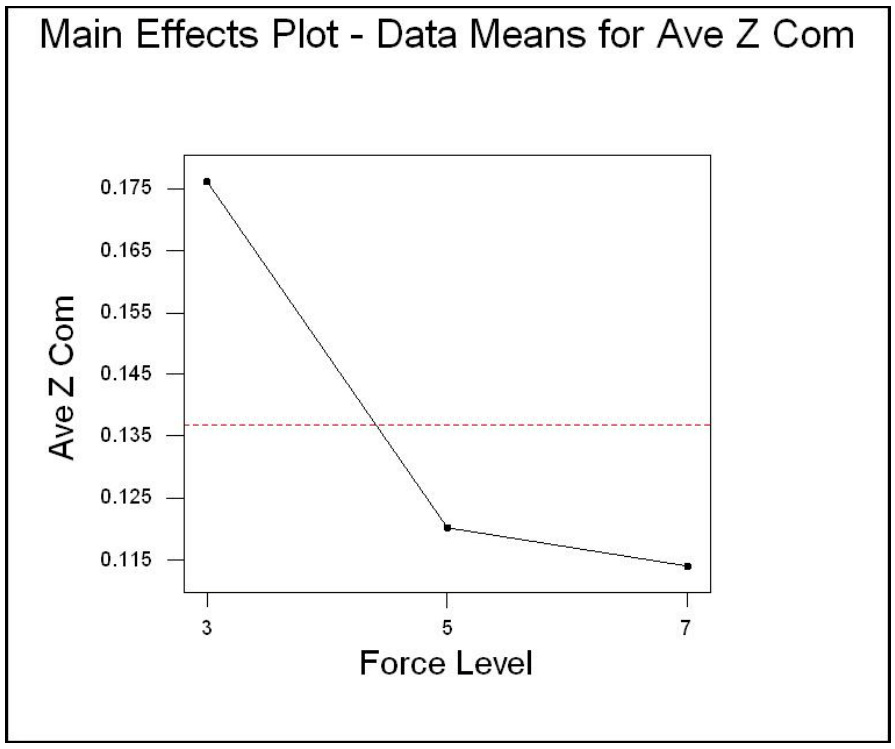

Figure 4-4: Main effect plot of the average commanded velocity in the z direction vs. force level

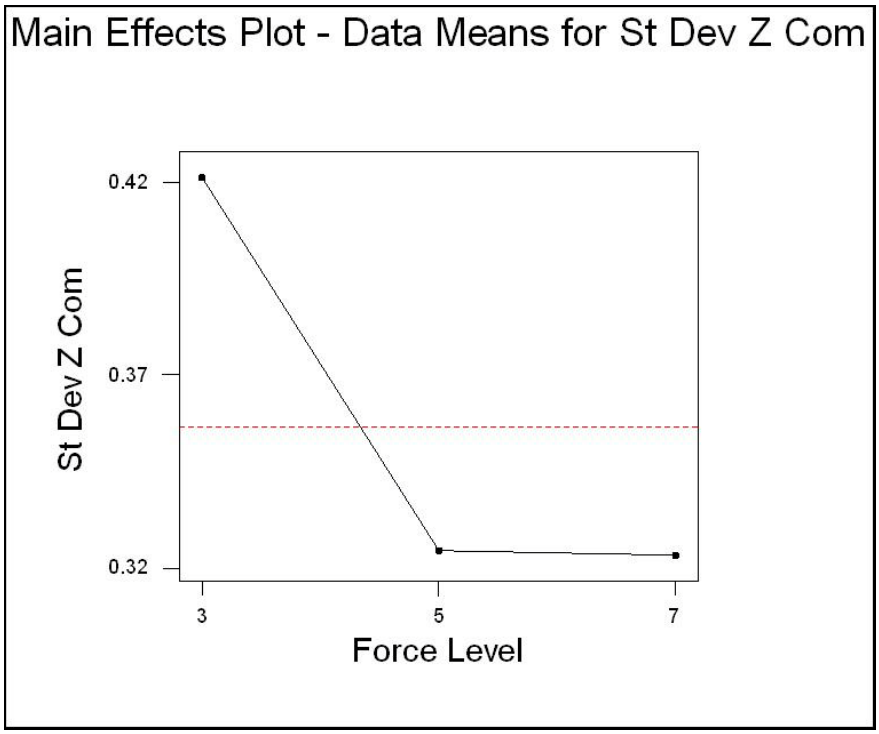

Figure 4-5: Main effect plot of the standard deviation of the z command direction vs. force level

The results of the force measures showed that higher force levels created higher average forces and greater standard deviations of forces, which again is a trivial outcome. The results also showed that the time to impact algorithm at the higher force levels had significantly larger force output than the other algorithms as shown in Figure 4-6. This figure indicates that the time to impact algorithm at the highest force level had the highest average force output. This is the same algorithm that showed a significantly lower hit length and the higher forces could be the cause. The results of the force measures in the separate directions will not be discussed because they are similar to the results for the composite forces.

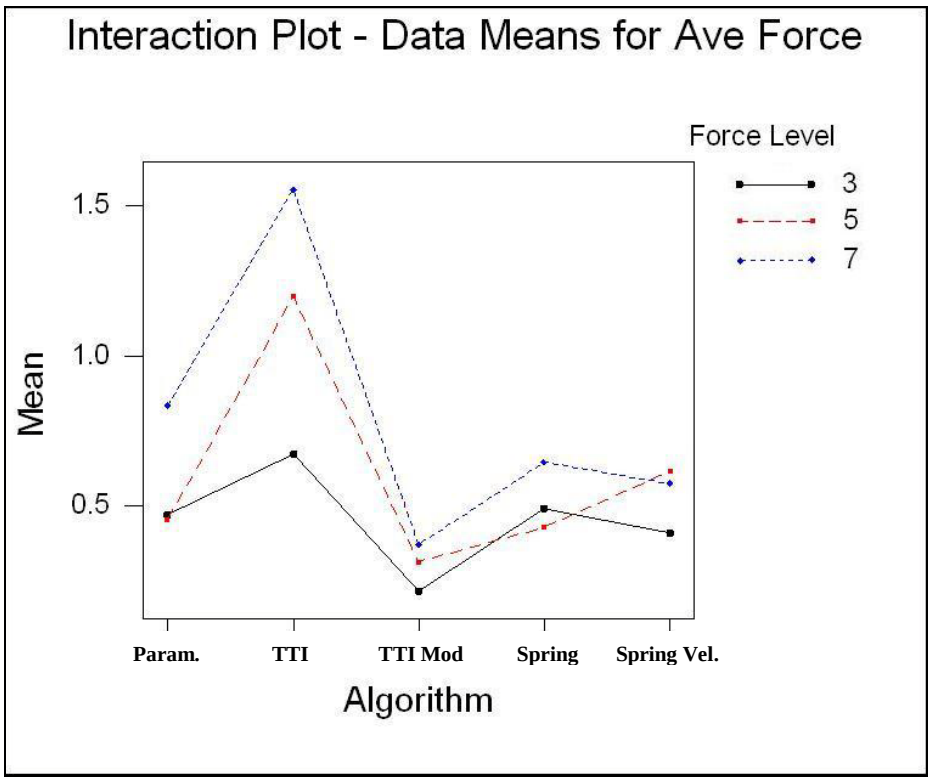

Figure 4-6: Interaction plot for average force output, algorithm, and force level

4.4.3 User Statistics and Post Experiment Questionnaire

All of the users for this experiment were current students at Brigham Young University. The average age was 23.67 years, ranging from 18 to 29. Twelve of the students were male and three female. Thirteen were right handed and two were left handed and all used their dominant hand for the experiment. The majority of the volunteers indicated that they had little or no experience with joysticks and flight simulators, some indicated moderate experience, and two indicated that they were very experienced.

After the experiment was completed, the users were asked to give subjective analysis of some of the aspects of the experiment. Each question was given a rating scale from one to five in increments of one. The results of the questionnaire will be briefly discussed.

When asked how much the force feedback influenced their flying, users responded with an average of 3.2, meaning it did influence them some, but not completely. They gave an average rating of 3.3 when asked how helpful the force feedback was and an average of 2.95 when asked if they felt more confident. More users could tell that there were different algorithms, with an average rating of 3.8, and users thought that the course was fairly hard, but not too hard, with an average rating of 3. The subjects indicated that the runs were mentally challenging, with an average rating of 4.1, and said that they would want force feedback to help them to fly a real quadrotor in which they had invested money, with an average rating of 3.8.

4.5 Experiment Conclusions

From the analysis of this experiment it can be concluded that higher forces resulted in fewer hits and that the time to impact algorithm at the highest force level was the only combination of factors that significantly reduced the hits when compared to having no force feedback, however it was not significantly better than any of the other combinations. None of the algorithms improved the time to completion, and the spring algorithms with velocity dependence created higher workload for users. These results indicate that, when commanding velocity, the time to impact algorithm is most likely the best and that the second spring algorithm is not desirable since it increases workload.

While the results of this experiment give some information about the effectiveness of the algorithms, they are not strongly compelling. The lack of significance between algorithms is likely due to high variation in the measures. It was observed that during the runs, users favored one objective (speed of completion or obstacle avoidance) over the other. This is one major source of variation that may have caused bias in the experiment.

5 Human Subject Experiment 2 – Angle Control

A second experiment was designed to test the effects of the force feedback algorithms as a pilot guides a quadrotor using angle commands from the controller, rather than velocity commands. As in the first experiment, the goal is to determine whether the algorithms are better than having no force feedback, and which algorithm is most beneficial.

5.1 Factors and Factor Levels

Information from the previous experiment was used in designing the second experiment. First, it was determined that the force level did not have a significant effect for forces greater than 5 N. For this reason, force level was not included as a factor in the experiment and the force was held constant for all runs. Secondly, it was decided to not include the spring algorithm with the velocity dependence because of its poor performance in the previous test. The time to impact algorithm with the no force zone was also not included as a factor level, not because it performed poorly, but because the original time to impact algorithm did very well and it was determined to be unnecessary to test more than one version of this algorithm.

A DOE was created with one factor, the force feedback algorithm, with four levels corresponding to (1) the dynamic parametric field, (2) time to impact, (3) basic spring algorithm, and (4) no force.

5.2 Design of Experiments

Because force level was not being tested as a factor for this experiment, the DOE setup was much simpler. The experiment was one factor with four levels. Each person who volunteered for the experiment completed one block of eight runs. Each algorithm was tested twice within the eight runs in random order. Eight people completed the experiment resulting in each algorithm being tested sixteen times.

The testing procedure was the same as the first experiment except that each person completed eight runs instead of six, after the two practice runs. Also, before the experimental runs the users were told that the primary objective was to get through the course without any collisions and the secondary objective was to do it quickly. This was changed from the previous experiment because controlling the quadrotor was more difficult in general with angle control. Also, in the previous experiment it was noted that not all users gave equal weighting to both objectives, which may have caused greater differences between users.

5.3 Results and Analysis

The measures for this experiment were the same as the previous experiment and were again analyzed using ANOVA with the block as a random variable. The ANOVA only indicates if there is a difference between the levels for a measure. When a measure was found to be significant, the algorithms were compared using a Tukey pairwise comparison test [19]. This test compares each level with all of the other factor levels to find out where differences occurred and whether they are statistically higher or lower. It is similar to the Dunnett test except that it compares all possible pairs of means to determine differences and uses a different calculation to

determine the test statistic and a different critical value for comparison. The results here are divided into direct and indirect measures, as described in Section 4.3.

5.3.1 Direct Measure Results

For the number of hits, the time to impact algorithm had significantly fewer hits than the other algorithms and no-force. No differences between the other algorithms were significant relative to the number of hits. For the length of hits, the time to impact algorithm was significantly lower than the spring algorithm and no-force. The parametric algorithm and no-force were lower than the spring algorithm. The box plot in Figure 5-1 shows the 95% confidence intervals for the different algorithms. In all of the figures in this section, 1 is the parametric algorithm, 2 is time to impact, 3 is the spring algorithm and 4 is no-force.

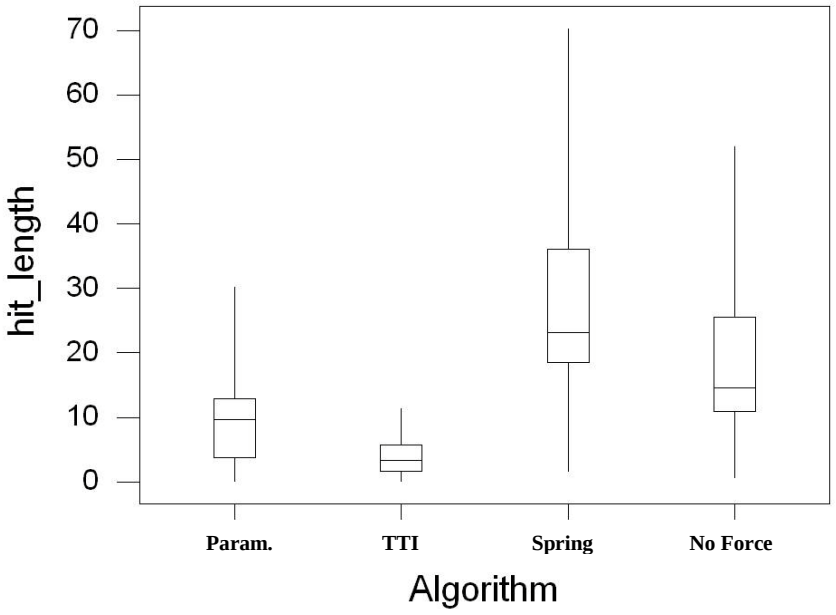

Figure 5-1: Boxplot for hit length.

No statistical difference was found for the amount of time taken to complete the course. This means that none of the algorithms decreased the time taken, but also that none of them increased the time. For workload, the spring algorithm caused a significant increase when compared to the parametric algorithm and time to impact. This is illustrated in Figure 5-2.

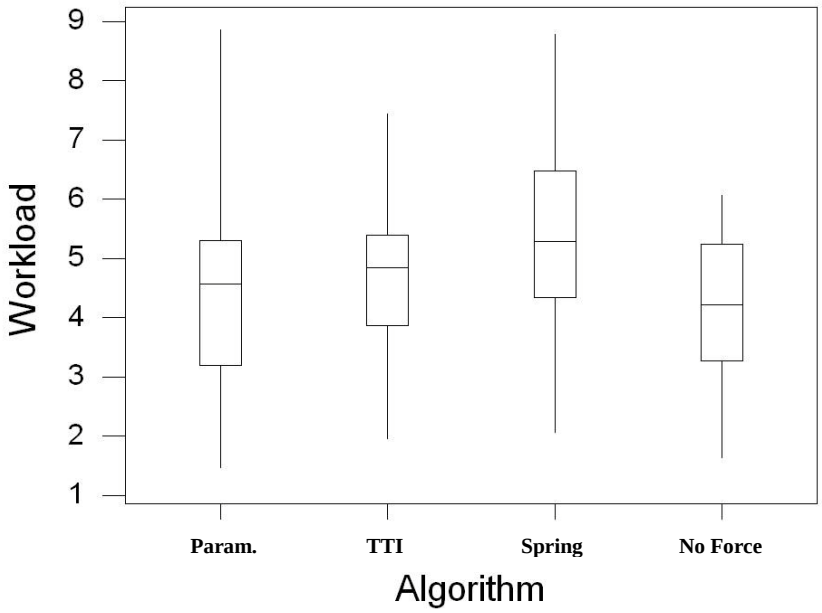

Figure 5-2: Boxplot for workload.

5.3.2 Indirect Measure Results

The indirect measures that are significant are the length, the average and standard deviation of velocity, and all of the force measures. None of the command measures had significant differences. The spring algorithm caused a significant increase in the path length when compared to the time to impact algorithm and no-force. For the average and standard deviation of velocity, both the parametric algorithm and time to impact showed significant decreases compared to the spring algorithm and no-force. The results for path length and

average velocity are shown in Figures 5-3 and 5-4. The fact that the parametric algorithm and time to impact both had lower average and standard deviation of velocity and performed better when comparing hits and hit length suggests that the lower velocity may help the quadrotor to avoid obstacles.

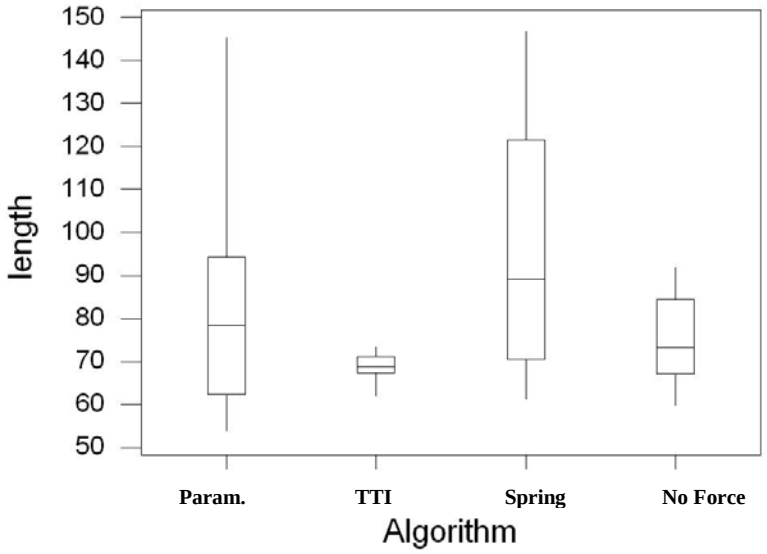

Figure 5-3: Boxplot for path length.

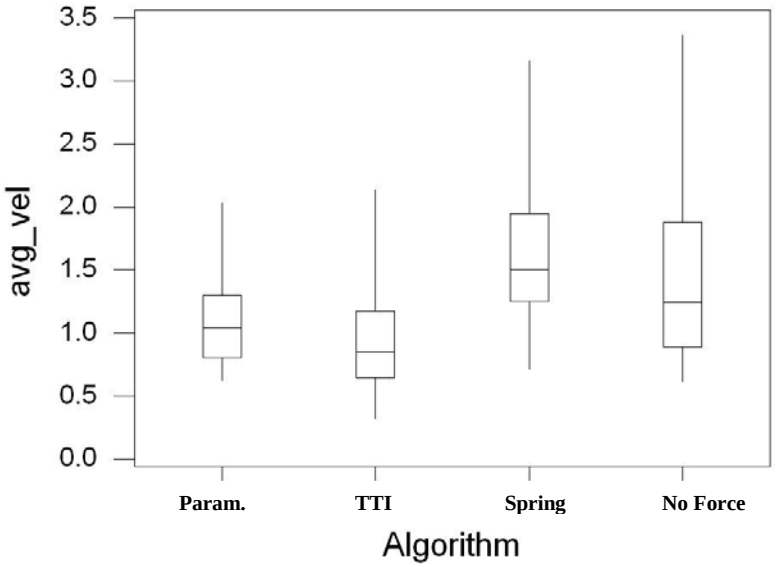

Figure 5-4: Boxplot for average velocity.

All of the force measures were significant. However, all of the results were similar for the separate directions, and so only the overall average will be discussed. All the algorithms naturally had a higher average force than no-force. The parametric algorithm had a lower average force than the spring algorithm and time to impact, and time to impact had a greater average force than the spring algorithm. This is shown in Figure 5-5. The reason no-force is not zero in this figure is because forces were still applied and recorded when an obstacle was hit. It is interesting to note that although the parametric algorithm and time to impact performed well in the hits measures, they had significantly different average force outputs. This means that it is not necessarily how much force is applied that causes fewer hits, but how the forces are applied.

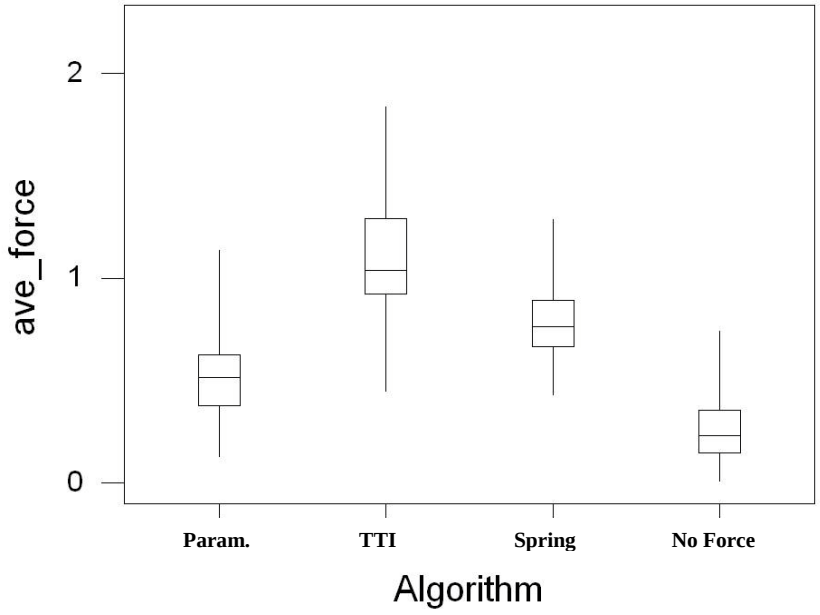

Figure 5-5: Boxplot for average force output

5.4 Experiment Conclusions

The experimental results suggest that the time to impact algorithm is the most useful to help pilots avoid collisions. It proved to be better than all the other algorithms in the number of hits and better than the spring algorithm and no-force in hit length. The parametric algorithm was better than the spring algorithm in hit length, but not better than having no force. Despite performing better in several key measures, the time to impact algorithm did not significantly increase workload or time to completion compared to the other algorithms. The spring algorithm is the worst, as it increased the hit length compared to no-force and increased the workload and path length.

To determine if piloting in velocity control or angle control was statistically better, identical runs (the same algorithm at the same force level) were taken from both human subject experiments and compared as if there was only one factor, the control mode, and two levels, velocity control and angle control. It was found that there are no significant differences for any of the direct measures, and the only indirect measures that are significantly different are the input device command average and standard deviation. These measures should be different because the control mode is different, with displacements of the input device corresponding to different commands to the quadrotor. This indicates that, although it is more intuitive, velocity control does not help users to avoid obstacles, do the task more quickly, or relieve the mental workload more than angle control. It was initially hypothesized that the lack of clear results in the velocity control mode may be due to the fact that it is a more natural and effective control mode, and therefore force feedback was unhelpful in successfully completing the course. The comparison of the two control modes suggests that this is not the case, since both control modes yielded similar direct measures.

6 Flight Tests

An experiment was conducted to demonstrate the use of the force feedback algorithms and software in piloting an actual quadrotor UAV in an indoor environment. In this experiment, the PHANTOM was used as the input and force feedback device, and ultrasonic sensors on the quadrotor provided information on the distance of the quadrotor from obstacles. The system is shown in Figure 6-1.

The PHANTOM positions are read into the ground station computer and converted into commands for the quadrotor. The program on the ground station computer is in MATLAB which communicates with an XBee radio over the serial port. These commands are sent to the autopilot on the quadrotor through an XBee radio that is mounted on the quadrotor. The autopilot sends its telemetry data back to the computer through the XBee radio. Simultaneously, the ultrasonic sensors read in range data and send them to the ground station computer through the XBee. The program on the ground station computer uses range information, telemetry information, and PHANTOM positions to calculate feedback forces, which are then applied to the pilot via the PHANTOM.

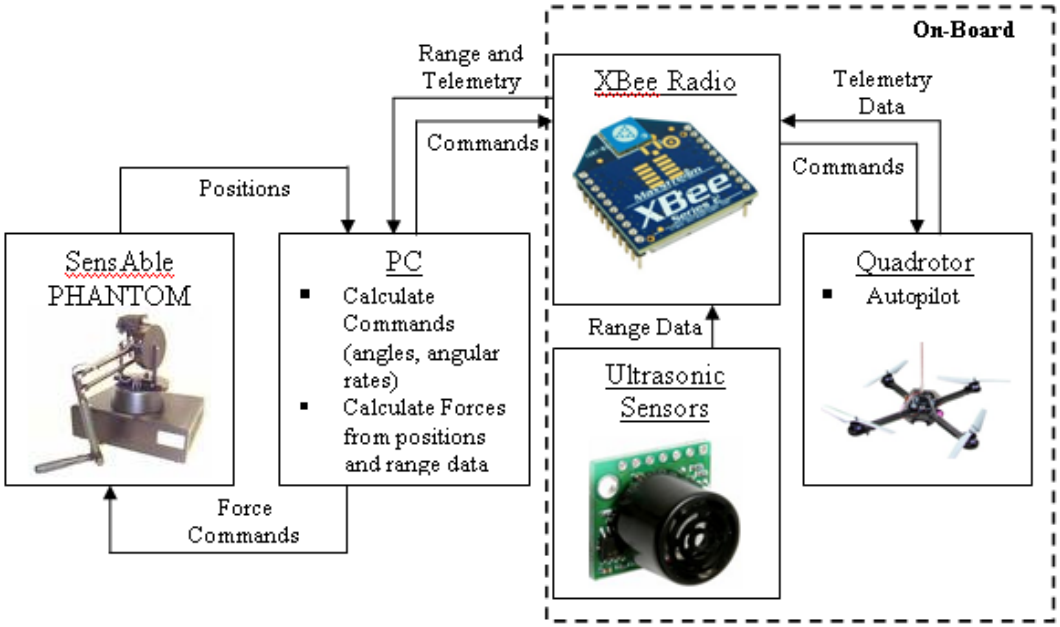

Figure 6-1: Overview of the main components and functions in the quadrotor system

6.1 The Hummingbird

The quadrotor used for testing is the Hummingbird developed by Ascending Technologies (Figure 6-2). The Hummingbird includes a central control unit, an autopilot, motor controllers and motors. The autopilot has two different control modes [20]. In one mode the autopilot controls the pitch and roll rates and in the other it controls the pitch and roll angles. In both modes the autopilot controls the yaw rate. When in angle control mode, the autopilot also has two modes for the height control. In one mode the autopilot controls the thrust and in the other mode it controls the rise and fall rates.

Figure 6-2: The Hummingbird by Ascending Technologies

6.2 Sensors and Communication

The sensors used to find distances of obstacles around the quadrotor are the MaxBotix LV-MaxSonar-EZ4 ultrasonic range finders. These range finders can detect objects at a distance of 0 to 254 inches with one inch resolution [21]. The EZ4 was chosen because it has the most narrow beam width of the EZ series. Three of these sensors were used for the testing; one pointing out the front of the quadrotor and one pointing out either side. The sensors sample every 50 ms and are sent as analog signals to the XBee radio, which converts them to digital signals for transmission to the ground station computer.

The XBee is a radio modem that operates at a 2.4 GHz frequency and has an indoor range of 100 ft and 300 ft for the XBee and the XBee-Pro, respectively [21]. An XBee-Pro is mounted

on the quadrotor and the range data from the ultrasonic sensors are sent to its analog-to-digital pins. The telemetry data from the quadrotor are sent to the digital out pin and the commands from the computer are sent to the digital in pin. All of this information is transmitted back and forth to an XBee connected to the serial communication port of the ground station computer. The information is read in and sent out of the serial port using Matlab.

6.3 Flight Testing

Flight tests were performed in a rectangular room without clutter using the system described and the time to impact algorithm. The tests were done with the autopilot in angle control and thrust control modes. During flight, the inputs from the PHANTOM controlled the pitch and roll angles, which allowed the pilot to control forward and lateral motion. The thrust and yaw angle were controlled independently to maintain a constant heading and altitude. Information was sent between the computer and the quadrotor at a rate of about 28 Hz. Each time data was sent, the commands from the phantom were sent to the quadrotor. However, only one packet of either the range information or the telemetry data could be sent to the computer. The packets were alternated so the information update rate in the computer program was about half the transfer rate. The pilot was in the room and had a clear view of the quadrotor at all times. The quadrotor was flown for several minutes and was purposefully brought close to the walls several times to generate large feedback forces. The quadrotor in flight is shown in Figures 6-3 and 6-4.

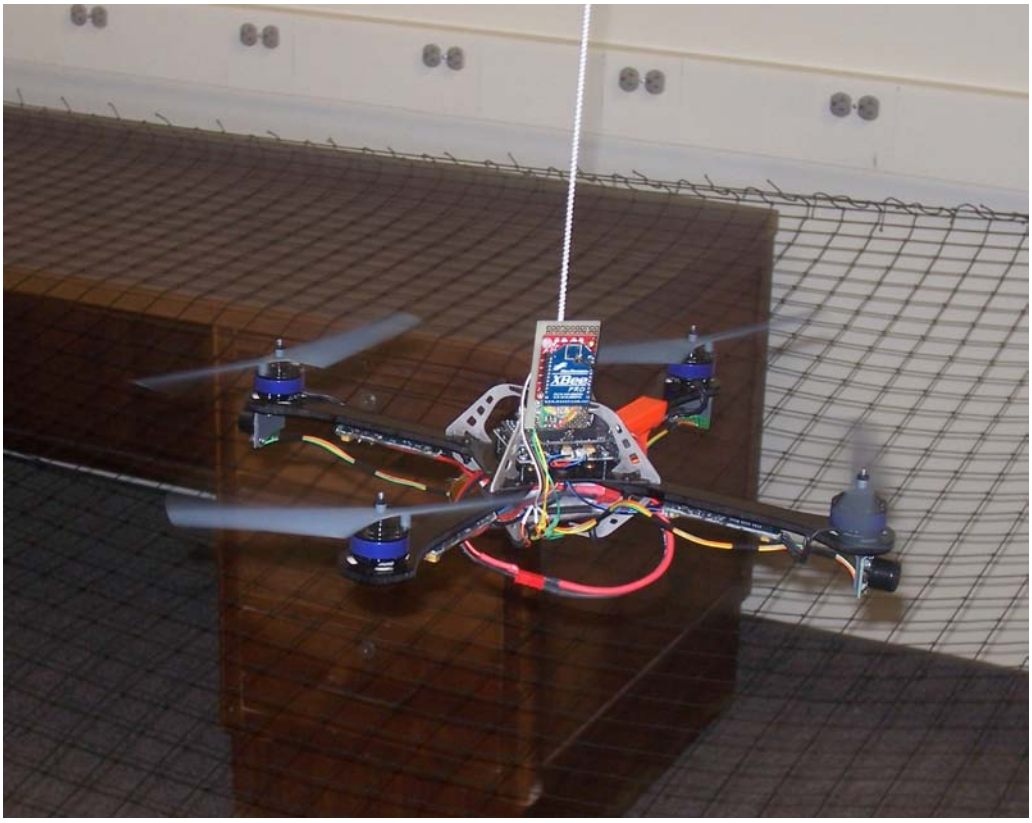

Figure 6-3: The quadrotor in flight with ultrasonic sensors and XBee radio mounted onboard

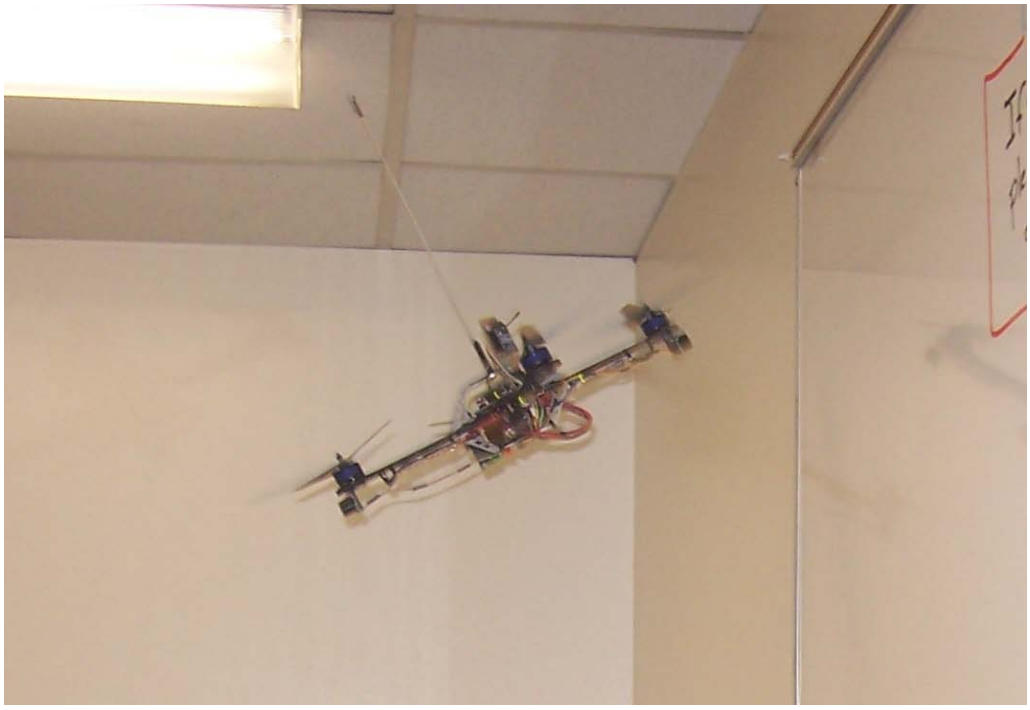

Figure 6-4: The quadrotor in flight near a wall to generate large feedback forces

6.4 Results

During each flight, the distance, velocity, and force information were recorded. Figure 6-5 shows a sample of the distance, velocity, and force for the forward direction for a portion of a flight. A negative value on the force plot corresponds to the PHANTOM generating a force in the backward direction to assist the pilot in avoiding the wall in front of the quadrotor. Because the force is dependent on both the velocity and the distance from the wall, the spikes in force output do not always occur when the quadrotor is close to a wall. In Figure 6-5 at around 3600 cycles, it can be seen that the force increase is caused by the distance coming close to zero. However, around 3850 cycles, the distance is not small, but there is a large increase in velocity.

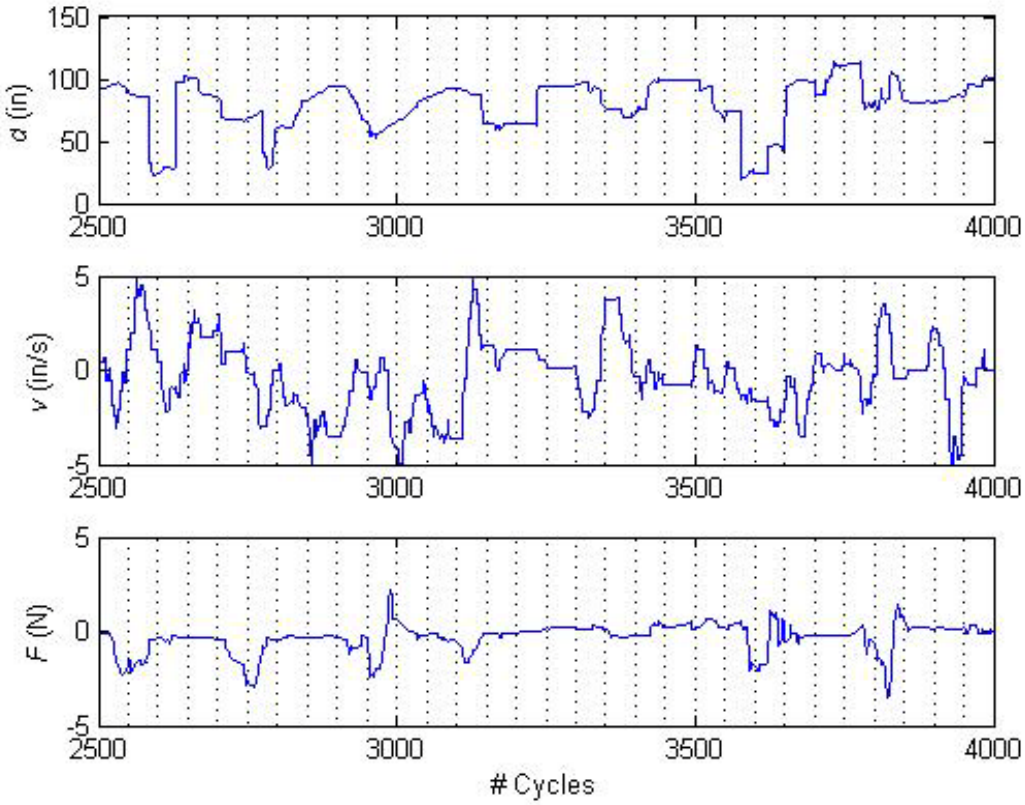

Figure 6-5: Distance, velocity, and force for the forward direction during a flight test

This can also be seen in Figure 6-6. This figure shows the flight path of the quadrotor as recorded from the forward and left side sensors. The arrows on the figure show the direction and magnitude of the force output at certain key points during the flight. The arrows near the top of the figure correspond to the output at cycle 3600, when the quadrotor is near the wall, but not moving very quickly. The arrows near the bottom of the figure correspond to the output at cycle 3850, when the quadrotor is not near the wall but was moving very quickly toward the wall. The arrows toward the left of the figure show a large force output in the lateral direction when the quadrotor was close to a wall on its left.

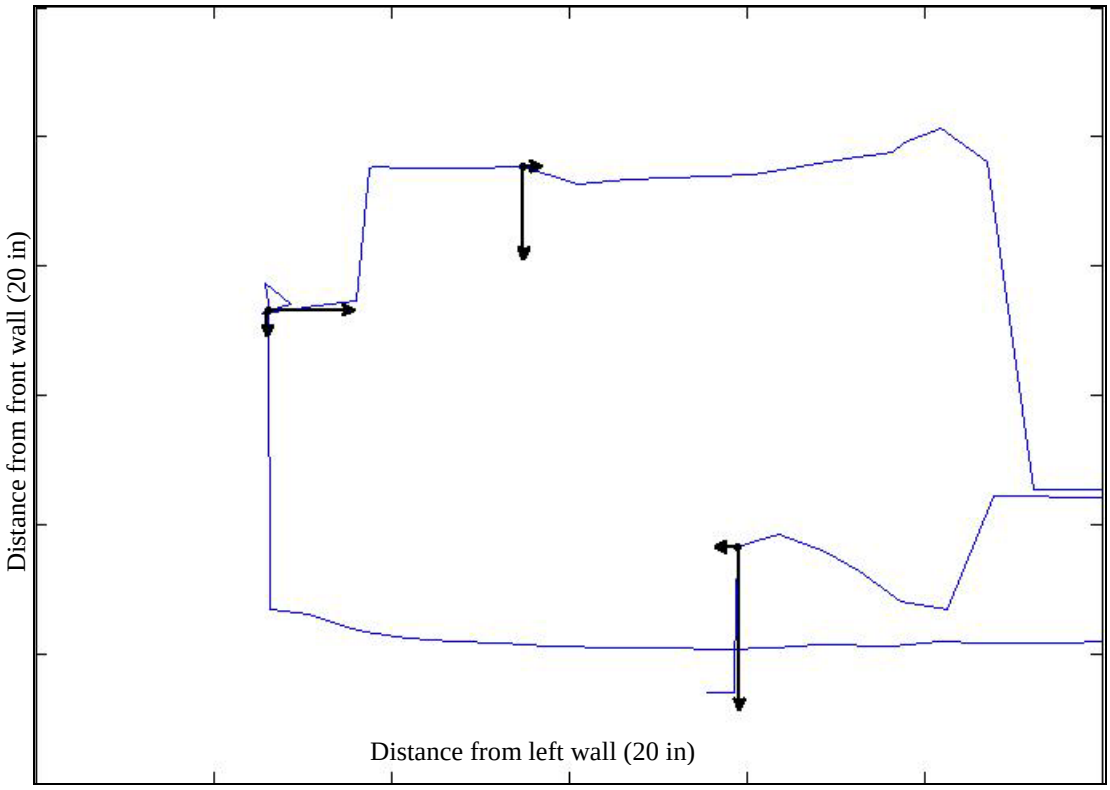

Figure 6-6: Flight path of the quadrotor in the room with arrows showing the direction and magnitude of the output forces at key points

The results from the flight tests show that the quadrotor was able to be controlled using the PHANTOM and that the forces were correctly generated as the quadrotor flew in an indoor environment. No quantitative measurements were made as to how well the forces worked, and different algorithms were not compared. The pilot noticed that appropriate forces were felt at certain times, but it was unclear if they helped to avoid hitting the walls. The tests only showed that correct forces were generated and that the pilot could feel and understand the forces.

7 Conclusions

In this thesis, force feedback was explored as a way to help pilots of remotely operated quadrotor unmanned aerial vehicles to avoid collisions while flying in indoor environments. Five force feedback algorithms were developed and tested in simulation with the PHANTOM haptic device, and user studies were done to compare the effectiveness of the algorithms at various maximum force levels. The system was also tested using an actual quadrotor flying in a rectangle room to demonstrate the functionality of the system.

7.1 User Studies

The results of the user studies suggest that some force feedback algorithms help pilots to avoid collision, whereas certain algorithms do not help, and others increase the number of collisions compared to no force feedback. The conclusions that are gained from the user studies are outlined as follows:

- In *velocity control mode* the time to impact algorithm significantly reduced the amount of collisions compared to having no force feedback. Although it was not statistically better than the other feedback algorithms, it may still be considered the best algorithm because the other algorithms were not statistically better than the no force feedback case. However, the results are not strongly compelling. The lack of significance between the algorithms is likely due to high variances in the measures, which make differences

difficult to discern. Furthermore, it was observed that users chose to favor one objective (speed of completion or obstacle avoidance) above the other, despite being instructed to give each objective equal weight. This may be an additional contributor to the high variability in the measures.

- In *angle control mode*, the time to impact algorithm is significantly better than other algorithms and the no force feedback case. The time to impact algorithm significantly reduced the number of collisions compared to all other algorithms without increasing the time taken to complete the task or the workload on the pilot.
- The time to impact algorithm performed best in both control modes and so it could be applied whichever mode is used in the future.
- There were no statistical differences between the control modes for the direct measures. This means that, although velocity control was thought to be more intuitive, it is not necessarily the preferred method of control. Angle control may actually be preferred because it requires simpler sensing capabilities and computation than velocity control and does not reduce the performance of pilots.
- Higher forces are typically more effective in decreasing the number of collisions compared to lower forces for the algorithms presented in this thesis.

7.2 Flight Tests

The flight tests described in this thesis demonstrate that the force feedback system may be used effectively to control the quadrotor with the PHANTOM, and that the system is capable of generating appropriate feedback forces. When the quadrotor was flown near walls or toward walls at high velocities, appropriate forces were output by the haptic device to suggest to the

pilot to slow the quadrotor or guide it away from the walls. The hardware developed for this work, including the wireless communication, ground station, sonar sensors, and haptic interface, were integrated and shown to meet the needs of a force feedback control system for quadrotor UAVs.

7.3 Future Work

In this thesis, multiple algorithms were developed and tested in simulation with user studies. Flight tests were performed with a physical quadrotor, and it was shown that this hardware system worked correctly. While the time to impact algorithm did help users to avoid collisions in simulation and the hardware system presented in this thesis did work for concept testing, there may better algorithms and systems that could be developed. This thesis is only a first step toward using force feedback to help the pilots of UAVs. There are several areas in which work should be pursued, including algorithms, experiments, hardware, and control. Future work could include:

- Develop and test additional force feedback algorithms. Time to impact proved to work well, but there are still other possibilities. Other algorithms might include vibration feedback or periodic force feedback. Also, other methods could be explored for eliminating instabilities caused by the algorithms instead of input device position dependency.
- Explore algorithms that use a combination of autonomous control and manual control. One possibility is to have mostly autonomous control with motion suggestions given by the pilot. Another possibility is to give most of the control to the pilot, and allow the

autonomous control to take over when danger is present to eliminate the possibility of collisions.

- Conduct user studies to compare control modes. Information was taken from the two user studies done for this thesis to compare control modes, but a user study should be designed specifically to test the differences between the modes.
- Conduct user studies using a physical quadrotor in an indoor course, rather than in simulation. When the hardware has been developed more an indoor course with safety precautions, such as nets or pads, could be created to do a user study.
- Investigate other commercial quadrotor platforms. The Hummingbird quadrotor used for this thesis is only one of many available commercial quadrotors. Others could have better computing, sensing, or payload carrying capabilities and should be explored.
- Explore the use of other force feedback devices (haptic interfaces, joysticks) to control the quadrotor. The PHANTOM worked well for this thesis, but it is an expensive device. There are a number of other commercial haptic devices available that could be used.
- Examine different sensors and different sensor layouts to gain range information. Ultrasonic sensors are cheap and light, but are not the most accurate or consistent range sensors available. Other instruments for finding range information could be explored as well as how the range finders could be placed on the quadrotor to give the most information.
- Develop accurate velocity estimates for use in velocity control mode. If velocity control is desired, accurate velocity estimates would be needed for use in the control loops. A good method for finding velocities would not only help the work in this thesis, but also other work dealing with UAVs

- Incorporate human sensing thresholds to determine optimal force feedback levels. Humans can only sense force above a certain threshold, and can only detect differences in forces with a certain resolution. This information could be studied to determine how to best apply force feedback.
- Develop a combined haptic/graphical interface to supply information to pilot in multiple modes, including visual indicators of obstacle vicinity. Haptics are not the only way to supply additional information to a pilot. Visual cues could be used as well, such as a red light on the screen, to warn pilots of close obstacles.

References

[1] D. D. Woods, J. Tittle, M. Feil, A. Roesler, “Envisioning human-robot coordination in future operations,” in *IEEE Transactions on Systems, Man and Cybernetics,* 2004, Vol. 34, pp. 210-218.

[2] M. Mulder, M. Mulder, M.M. van Paassen, S.Kitazaki, S. Hijikata, E.R. Boer, “Car-Following Support with Haptic Gas Pedal Feedback,” Proceedings of IFAC HMS Symposium, Atlanta, GA, 7-9 September, 2004.

[3] M. Mulder, D.A. Abbink, E.R. Boer, “The Effect of Haptic Guidance on Curve Negotiation Behavior of Young, Experienced Drivers,” in *IEEE International Conference on Systems Man and Cybernetics*, 12-15 October 2008, pp. 804-809.

[4] K. Yano, A. Takemoto, K. Terashima, "Semi-automatic obstacle avoidance control for operation support system with haptic joystick," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2-6 Aug. 2005, pp. 1229-1234.

[5] M. Frey, D.E. Johnson, J. Hollerbach, "Full-Arm Haptics in an Accessibility Task," in *symposium on Haptic interfaces for virtual environment and teleoperator* systems, 13-14 March 2008, pp.405-412,

[6] S. Charoenseang, A. Srikaew, D. Wilkes, K. Kawamura. “Integrating Visual Feedback and force feedback in 3-D Collision Avoidance for a Dual-Arm Humanoid Robot”, in *International Conference on Systems, Man and Cybernetics*, 11-14 Oct 1998, pp.3406-3411.

[7] J. Borenstein, Y. Koren, “Real-time obstacle avoidance for fast mobile robots in cluttered environments,” in *IEEE International Conference on Robotics and Automation*, 13-18 May 1990, pp.572-577.

[8] S. Lee; G.S. Sukhatme, G.J. Kim, C. Park, “Haptic control of a mobile robot: a user study,” in *IEEE/RSJ International Conference on Intelligent Robots and System, 2002*, vol.3, pp. 2867-2874.

[9] S. Hong, J. Lee, S. Kim, “Generating artificial force for feedback control of teleoperated mobile robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999, vol.3, pp.1721-1726.

[10] D. P. Barnes and M. S. Counsell, “Haptic Communication for Remote Mobile Manipulator Robot Operations,” in Proc. of *81h American Nuclear Society Topical Meeting an Robotics* & *Remote Systems,* April 1999.

[11] T. M. Lam, M. Mulder, and M. M. Van Paassen, “Collision Avoidance in UAV Tele-operation with Time Delay,” in *IEEE International Conference on Systems, Man and Cybernetics*, 7-10 October 2007, pp. 997-1002.

[12] T. M. Lam, V. D'Ameilo, M. Mulder, and M. M. Van Paassen, “UAV Tele-operation using Haptics with a Degraded Visual Interface,” in *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 3, 8-11 October 2006, pp. 2440-2445.

[13] H. W. Boschloo, T. M. Lam, M. Mulder, and M. M. Van Paassen, “Collision avoidance for a remotely-operated helicopter using haptic feedback,” in *IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands, 10-13 October 2004, pp. 229-235.

[14] T. M. Lam, M. Mulder, and M. M. Van Paassen, “Haptic Interface for UAV collision avoidance,” *International Journal of Aviation Psychology*, Vol. 17, No. 2, 2007, pp. 167-195.

[15] S.G. Hard and L.E. Staveland, “ Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research,” in *Human Mental Workload*, P.A. Hancock and N. Meshkati, Eds., North-Holland, Amsterdam, The Netherlands, 1988, pp 139 – 183.

[16] V.E. Balas, M.M. Balas, "Driver Assisting by Inverse Time to Collision," in *Automation Congress, 2006. WAC '06. World*, 24-26 July 2006, pp.1-6.

[17] “Specifications Comparison for the PHANTOM Premium 1.0, 1.5, 1.5 High Force and 3.0 Haptic Devices,” Jan. 7, 2009. [Online]. Available: http://www.sensable.com/documents/documents/STI_Jan2009_PremiumModelsComparison_print.pdf. [Accessed: Aug. 5, 2009].

[18] R. W. Beard, “Quadrotor Dynamics and Control,” Brigham Young University, Tech. Rep., 2008. [Online]. Available: https://contentdm.lib.byu.edu/cgibin/showfile.exe?CISOROOT=/EER&CISOPTR=217&filename=218.pdf#search=%22Beard%22. [Accessed : Aug. 5, 2009]

[19] “SAS/STAT 9.2 Users Guide,” 2008. pp 2516-2517, [Online]. Available: http://support.sas.com/documentation/cdl/en/statug/59654/PDF/default/statug.pdf. [Accessed: Aug. 7, 2009].

[20] “X-3D-BL Scientific User’s Manual,” [Online]. Available: http://www.asctec.de/main/updates/x3d_scientific_v30_manual_en.pdf. [Accessed: Aug. 5, 2009].

[21] “LV-MaxSonar-EZ4 High Performance Sonar Range Finder,” Jan. 2007. [Online]. Available: http://www.maxbotix.com/uploads/LV-MaxSonar-EZ4-Datasheet.pdf. [Accessed: Aug. 5, 2009].

[22] “XBee Multipoint RF Modules,” 2008. [Online]. Available: http://www.digi.com/pdf/ds_xbeemultipointmodules.pdf. [Accessed: Aug. 5, 2009].