Fuzzy Control of a Robotic Blimp

Laith M. Alkurdi

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2011

Abstract

The robotic blimp platform that the School of Informatics possesses presents a challenging and exciting robotic control problem. The lack of a mathematical model for the blimp means no computer simulation methods can be used to tune parameters of simple controllers or simulate the results of more complex controllers. The constant existence of environmental disturbances and model uncertainties means that a truly robust controller is needed for the robotic blimp. This dissertation provides three different controllers and compares their performance. The controllers are two state based controllers, and a fuzzy logic controller. While most projects - that apply fuzzy logic control on a robotic blimp - discussed in the literature rely on computer simulation, we applied fuzzy logic control on a physical blimp and studied how the blimp actually navigated in space using the three controllers. Results showed that the fuzzy logic controller performed better than the state based controllers.

Acknowledgements

Firstly, I would like to thank my supervisor Robert Fisher for his helpful discussions and support. Furthermore, thanks to John Hussey and Antonios Ntelidakis for their work on the blimp software and the earlier controller algorithms.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Laith M. Alkurdi*)

This dissertation is dedicated to my parents and brothers for all their love and support.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

A blimp is a special kind of lighter-than-air airships; it does not have a rigid skeleton supporting its balloon. Blimps and airships automation have recently emerged as an attractive field of research due to their properties.

Unmanned aerial vehicles (UAV) in general have advantages over unmanned ground vehicles. They are able to reach locations where it is hard for ground vehicles to reach due to hazards or terrain limitations. They also have the advantage of a larger field of view making them able to survey and collect data of a larger area of terrain at a given instance.

Blimps also have advantages over winged unmanned aerial vehicles and helicopters. Blimps have much safer failure degradation. They are able to hover over one area for a long time, achieve low altitude flights and do not suffer from maneuverability constraints. They also have minimal vibration and do not influence the environment they are in. The properties previously mentioned make them ideal for data collection, exploration, monitoring and research applications. They take off and land vertically, this means that they can be easily deployed with no need for a runway; which make them attractive as platforms for rescue operations or as communication beacons when communication is cut-off from a certain area. Other attractive properties include long flight durations, low energy consumption as they depend on buoyancy to achieve vertical position.

Blimps have been studied as a viable platform for communication platform that could be deployed rapidly [5], advertisements and atmospheric data collection and analysis. They are also attractive for military operations such as surveillance and rapid equipment deployment. Blimps serve as an option for providing images and information of regions which suffered natural catastrophes. They have also been used in mine

detection [32]. Map building and localization of targets have been also been studied through the work of LASS/CNRS [10, 21, 11, 9]. Astro-explorations are also an application studied by the Jet Propulsion laboratory at NASA [15], [30].

This work compares three controllers applied to the robotic blimp, a 3-parameter state based controller, a 4-parameter state based controller and a fuzzy logic controller. The three controllers were built to be effective and robust to environmental disturbances, in the absence of a mathematical model for the robotic blimp. Several performance measures have been formulated for comparison, and results have shown that the fuzzy logic controller has been the better performing controller.

1.1 Blimp used

The blimp employed in this project is the Surveyor blimp YARB (Yet Another Robotic Blimp) which is a 66" helium blimp. The onboard electronics include a Blackfin processor, color camera, Matchport wireless LAN interface. This robotic blimp is driven by three motors, two propellers and a third vectoring motor. The blimp's motors can take a range of [0,128] for forward motion, where 0 is 0% of maximum output and 128 is 100% of maximum output. The blimp used is shown in figure 1.1.

Figure 1.1: The blimp used

The blimp is 1.68m long and has a maximum diameter of 0.76m giving it a fineness ratio (length/diameter) of 2.2. It has a volume of $0.26m^3$ and a total lift capacity of 0.3kg given that the lighter than air gas used is helium. While hydrogen is a cheaper alternative that provides more lift capacity for the same volume, helium remains the safer choice.

The blimp platform under study has a few drawbacks making its control rather challenging. The most challenging aspects of the control problem are modeling the dynamics of the blimp and accounting for uncertainties. Examples of uncertainties include disturbances in the form of temperature and pressure variations that could vary the size of the blimp's envelope and vary the buoyancy, other disturbances include wind gusts. Another problem faced in this project is that the blimp's envelope leaks helium varying its buoyancy from one test run to the other. Airship dynamics are also notoriously hard to control due to a large moment of inertia [16]. The blimp's lack of an internal rigid frame structure makes its envelope susceptible to expansion and contraction due to air pressure and temperature variations, adding uncertainty to the blimp's dynamic model. Signal latency has also been observed in our platform as well as delay in control signals. Blimps have another disadvantage of limited payload. Payload, when speaking about blimps, is a function of envelope size, and with small blimps limited payload means a limited amount of sensors the blimp can be equipped with. That means that the variety of information that the blimp's controller can be fed is limited.

This project comes as a continuation to the work of Antonios Ntelidakis' MSc dissertation 'Using a blimp to model an interior model of the forum' [25] and the work of Robert Fisher and John Hussey who later worked on the development and implementation of control algorithms. Under the existing controller the system converges to the desired position, however; it takes a rather long settling time and rarely takes a straight line from the initial to final position. Furthermore, the system is very susceptible to disturbance in the form of air gusts and temperature/pressure variations. In this work additional control policies and control algorithms are to be studied and applied in order to enhance the performance of the overall system.

1.2 Project objective and outcomes

The aim of this project is to improve the performance of the robotic blimp platform whose three control parameters (height from the ground, orientation towards the goal,

and distance to goal) are under constant random environmental disturbances. Performance is calculated with respect to the blimp's ability to maintain a heading and travel between two points. Further detail on performance metrics is described in chapter 3 of this document. To that end different controllers are applied to the blimp and the weaknesses and strengths of each controller are studied. The project outcomes are listed as follows:

1. We formulated a set of performance measures as a benchmark for blimp controller testing.
2. We obtained a set of performance measures for the 3-parameter state controller and identified the drawbacks and failures.
3. We studied the failures of the 3-parameter state controller and improved it to obtain a 4-parameter state controller.
4. We obtained a set of performance measures for the 4-parameter state controller and identified the drawbacks and failures.
5. We built a fuzzy logic controller and tested it on the blimp.
6. We obtained performance measures from the fuzzy logic controller.
7. We fine tuned the parameters of the fuzzy logic controller.
8. We compared the performance of the fuzzy logic controller to the two previous controllers.

The rest of this dissertation is organized as follows. Chapter two presents major previous works on the proposed controller. Chapter three discusses the methodology of testing and the performance measures used. Chapter four introduces and discusses the design and the results of the 3-parameter state controller. Chapter five introduces and discusses the design and the results of the 4-parameter state controller. Chapter six introduces and discusses the build and the results of the fuzzy logic controller. Finally the three controllers are compared against each other in chapter seven. Future work is also discussed in chapter 7.

Chapter 2

Background

This chapter introduces background to work done on airships and blimps. The first section mentions the major airship platforms and the control algorithms used. The second section focuses on fuzzy logic control schemes applied on airships and blimps.

2.1 Airships in the literature

This section aims to provide a summary of major airship platforms and discusses the control aspects used in each of these projects. Reviews on airship platforms as well as other UAV can be found in [1, 23, 26].

The University of Stuttgart's project "Lotte" is a 15m airship, with a volume of $107m^3$ and has a maximum payload of 12kgs. It has been a platform for many research projects such as aerodynamic research [24]. The dynamical characteristics of the ship have been modeled using system identification techniques [18]. The control relies on a number of sensors input such as GPS (global positioning system) information for position tracking as well as electronic compasses. The accelerations are calculated using inertial measuring units and the helium temperature and pressure is calculated and compensated against, a full description of the sensors used in this project can be found in [20] and a discussion on the controllers used is given in [31].

The LAAS-CNRS airship "Karma" is 8m, has a volume of $15m^3$and a maximum payload of 3.5kg [12]. This platform had been developed for high resolution terrain mapping and the controllers are built to execute planned trajectories by using the blimp sensor input and detecting special ground elements [22]. Positioning is done through vision, where two successive frames are analyzed to get a position update. The controller assumes decoupling between longitudinal and lateral planes. Once the airship

achieves desired longitudinal position the lateral controller starts to achieve path following. The airship's control algorithm involves geometric and dynamic models whose constraints are taken into account by employing backstepping techniques [10]. This airship platform has been used to apply SLAM (simultaneous localization and mapping) techniques successfully as discussed in [12].

The Titan Areobot project developed at the NASA and the Jet Propulsion Laboratory at the University of California was proposed to be used for planetary exploration on Titan, one of Saturn's moons [4]. The airship is an Airspeed AS-800B, it utilizes a nonlinear airship model for control purposes discussed in [27]. The controllers were built to accomplish tasks such as loiter, hover and cruise. A special controller is built for subtasks of ascent, descent, turning and altitude control. A full list of controllers that include sequential-loop-closure and linear-quadratic-regulator control algorithms is discussed in [19].

The AURORA Airship project at the Autonomous Institute of CTI Campinas, Brazil, is another important airship platform that has been used for environmental monitoring missions, investigations of airship dynamic models and visual servoed guidance[2]. The control of this airship is discussed in [28] where the airship makes use of a proportional integral (PI) controller and a proportional derivative (PD) controller to follow a path trajectory by outputting a heading angle. The problem of hover control has been also been investigated using this platform and is discussed in [3] where image processing is used to provide an offset from the desired position which then is input into the controller to account for the positional deviation.

2.2 Blimp fuzzy logic control

This section introduces major work done on proportional integral derivative (PID) and fuzzy controllers applied to blimps. While the main focus will be on projects relating to fuzzy controllers, we will mention ongoing work in the form of intelligent controllers such as model predictive controllers and reinforcement learning controllers that have shown very good results when applied to blimps.

Acquiring the blimp dynamical model is a first step of studying controller design. A general dynamical model is presented in [6]. In this work, a platform for controller design and simulation research is presented through a complete physical and dynamical model of the blimp.

Classical control methods in the form of PID control have the advantage of simple

implementation and reliability, however it can be computationally expensive to model the system and tune its parameters. Work on a PD controller can be seen in [2]. This project employed a dynamical model controlled by a PD error controller that gets feedback from an onboard camera that sends feedback signals to the controller. PID control has been also been applied to landing of a blimp by Toshihiko Takaya in [29], using orbital control. Another platform is presented in[11] where a PID controller is used for altitude and horizontal positions of the blimp.

In the work of Falahpour et al. in [5], a fuzzy logic controller was compared with a PID controller. System model and dynamics were derived in order to apply PID control. The model accounted for air friction and random wind gusts. The PID controller was able to achieve the desired position and could cope with gusts of wind of varying direction and force. The fuzzy controller used three membership functions for the inputs and five membership functions for the outputs. There are 4 error inputs (plane position, orientation and angular speed) giving 81 control rules. The defuzzification method used was the weighted average method. Results from the comparison showed better performance with the fuzzy controller in terms of less oscillation and faster convergence speed. This result was obtained under MATLAB simulation of the second order balloon dynamic model.

Gonzalez et al. in [7] apply a fuzzy altitude controller on a low-cost autonomous indoor blimp and compared it to PID control. Vertical and horizontal controllers were decoupled much like our system. The system also employs a fuzzy collision avoidance controller where a PID collision avoidance controller failed to provide satisfactory results. PID altitude control parameters were experimentally calculated using the Zieger-Nichols method and showed good results in stable, undisturbed environments but showed large oscillations in environments with disturbances. The fuzzy logic controller has two inputs of velocity and vertical position error and employed five member ship functions for each input, actuation output had nine membership functions. The Fuzzy controller out-performed the PID controller in practical tests especially in environments with wind disturbances. The same platform had a fuzzy collision controller that uses five membership functions for velocity and three membership functions for positional error. The fuzzy controller showed the desired behavior while the PID had oscillatory behavior and was judged to be inadequate.

In [14] altitude control of an autonomous airship is investigated with the use of fuzzy logic. Seven triangular membership functions were used for the positional error as well as for the speed of the blimp. The controller has two fuzzy logic subsystems,

one that calculates the current error and the other calculates the predicted error. Each one is activated depending on the blimp's altitude. This compound controller was designed in this way to be robust to disturbances. The results showed that the blimp was able to achieve and maintain the required altitude as well as being robust to parametric perturbations and disturbances.

Backstepping control, model-predictive control and reinforcement learning control of autonomous blimp navigation are prominent control methodologies currently receiving much attention. Important reviews and introductory material to the field for control of autonomous airships is given in [23]and [26].

Our work in this dissertation is different from the previously reviewed projects. While most projects reviewed rely on a mathematical model and computer simulation, this dissertation applies fuzzy logic control as well as state controllers to a physical blimp. This allows us to realistically understand the performance of the robotic blimp as well as the disturbances that affect its operation. This dissertation applies fuzzy logic control to the position and orientation of the robotic blimp, while literature focuses on fuzzy logic altitude control.

Chapter 3

Methodology and Setup

This chapter discusses the experimental setup used in this dissertation. Methods to measure the performance of the controllers are also discussed. Finally, problematic issues and sources of errors are mentioned.

3.1 Introduction

The blimp platform has proved to be a tough platform to work with, and its control is a very challenging task. The lack of a mathematical model meant that computer simulation was not available and testing for different parameters had to be done physically with the blimp. This becomes a tedious task of gathering many test runs for the same parameters. Many test runs with the same settings are important to rule out randomness and noise. Each run averages around 3.5 minutes. The testing is limited within the hours of 5pm to 9pm inside the Informatics Forum depending on external lighting. The blimp cannot be tested before 5pm as the air condition systems would be still on in the testing area, an environment where the blimp would not be able to function at all. Dependency on lighting conditions will be discussed later in this chapter.

3.2 Experimental setup

In this section, the procedure for setting up the blimp for experimentation as well as the experiment itself will be discussed. Expected results are also presented.

The blimp is a combination of two parts, the balloon and the gondola. They are attached via Velcro at the beginning of each experimentation day and removed at the end of the day. Before experimentation begins, the blimp is topped up with a small

amount of helium to attain a natural buoyancy of 15 feet with the gondola attached; this is because the blimp leaks helium overnight.

When the blimp is taken out into the testing area, it is placed at the position shown in figure 3.1 facing the point (39, 17) feet and is expected to travel to point (19, 17) feet in the (x,z) plane. The y position, which represents the height of the blimp from the ground, is expected to be in the 15 feet range throughout the entire experiment. A run is defined by a start at the point (39, 17) a forward motion towards (19, 17) a 180 degree turn at (19, 17) and a forward motion towards (39, 17). The blimp is said to have reached a target point ((39, 17) or (19, 17)) if it is within a 3 feet range of it. Figure 3.2 illustrates the blimp's test run in 3D (3 dimensions). The (X,Y,Z) position of the blimp is recorded at each point of the path.

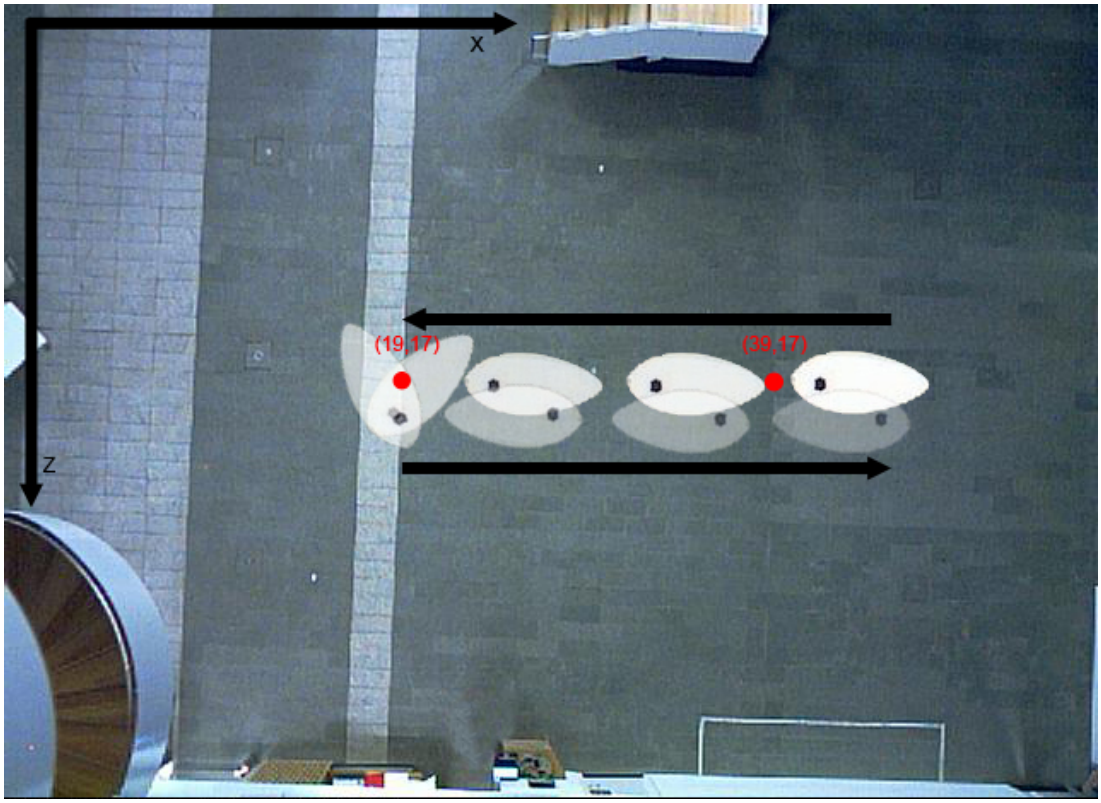

Figure 3.1: Experimental run

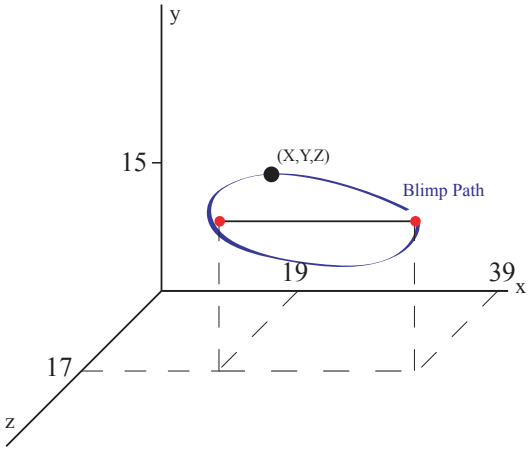

Figure 3.2: Experimental run in 3D

The software used for the experiments uses blob detection algorithms to detect the blimp. The center of mass of the blimp is recorded to obtain the (x,z) position. The height of the blimp is calculated by looking at the area of the blimp and estimating the distance from the ground. As this is done every frame and the lighting conditions are changing, the height readings (y coordinate) are usually rather noisy. The time between each position update is calculated and later is used to obtain a measure of time taken for each run.

The recorded data from the runs are then used to plot four important graphs. The graphs shown below are for a perfect controller. The first graph is the blimp's (x) position against motor commands as shown in figure 3.3. The second graph is the blimp's (z) position against motor commands as shown in figure 3.4. The third graph is the blimp's (y) position or height against motor commands as shown in figure 3.5. Finally figure 3.6 is the blimp's run as seen from the top view ((x,z) plane), this can be thought as a lap between two points. All the runs were normalized to 250 motor commands for comparison.

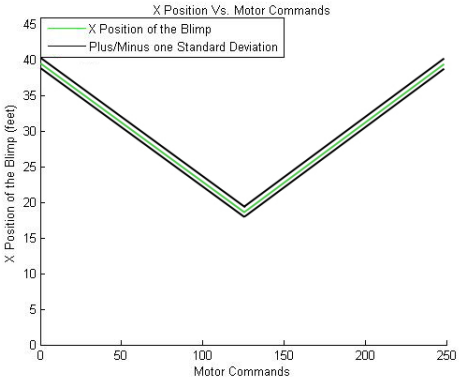

Figure 3.3: X position vs. Motor commands

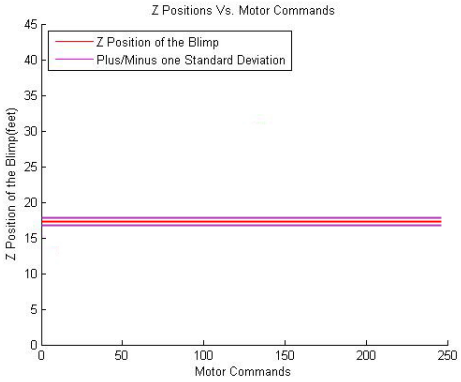

Figure 3.4: Z position vs. Motor commands

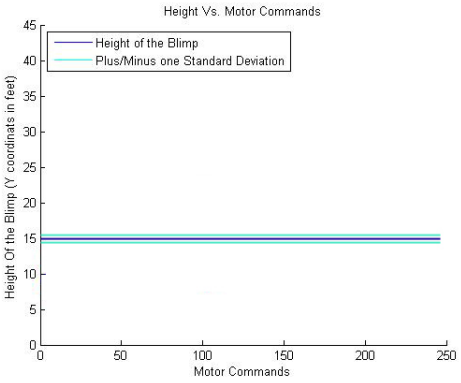

Figure 3.5: Height vs. Motor commands

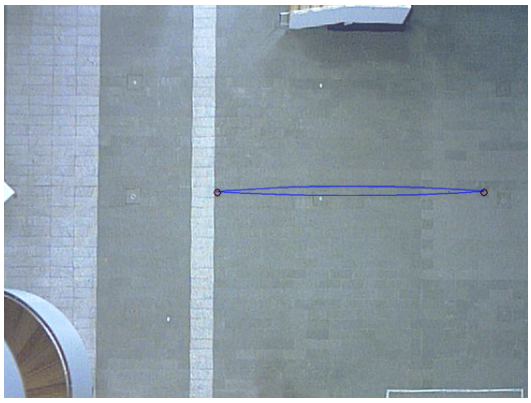

Figure 3.6: Z position vs. X position

The focus in this work is on the (x,z) plane, as the three controllers investigated in this work are built to maneuver the blimp in this plane. In the case that the blimp goes below or above a four feet threshold of the desired height (15 feet), height regulation is required through the use of a height controller. This height controller is applied with no regard to the (x,z) position. Thus given the disturbances the blimp can undergo, it may encounter drastic situations where it plunges to low heights and requires the controller to be applied. When this happens the blimp will gain altitude until the required height is obtained. During this controller action, the blimp (x,z) position may be be altered. The (x,z) position is not usually maintained during the height controller action because the movement at that stage is rarely vertical as the blimp would be moving with a certain momentum and will keep drifting in that position as it gains altitude. This tampering

with the blimps (x,z) position during altitude control may give false readings to how the controllers work. Thus figure 3.5 (height vs. motor commands) becomes important to understand the goodness of the various test runs, and a measure for comparability between different controllers.

Each controller was tested for 20 runs, 10 of them were run under notable disturbances in form of multiple people passing and/or elevators being called at time of run. The remaining 10 were run without these disturbances. A run is only terminated and discarded if the blimp exits the frame that is shown in figure 3.1. Each controller will be studied against itself, by taking the average of each of the 10 runs with and without disturbances. This is to understand the robustness of the controller to noise and environmental disturbances. Then different controllers will be studied against each other by taking the average of the 20 runs taken under the same controller. The next section discusses how controller analysis is performed once these runs are obtained.

3.3 Performance measures

This section discusses the performance measures that are extracted from the runs of each controller. The method of calculation as well as the significance of each of these measures is also given.

The first measure of performance is obtained by plotting the averages of the runs and plotting them as figures 3.3 through 3.6 given in the previous section. We would like them to be as close as possible as the plots shown.

The second measure of performance is obtained by calculating the area between one standard deviation of the means on each of the four plots; this can be thought of as a measure of repeatability for the controller. A large area is an indication of the chaotic nature of the controller and can be taken as an indication of unreliability.

The third measure of performance is calculated by looking at the average time required to finish a lap by a controller. If speed and accuracy can be combined then this will be a measure of a good controller however it may be the case that time is a tradeoff for a better performing controller.

The fourth measure of performance is calculated by taking the distance of each recorded 3D data point (x,y,z) indicating the location of the blimp in space and calculating the perpendicular distance from a 3 dimensional line passing through the two points the blimp flies between, (39,15,17) and (19,15,17). If these distances are added along the entire lap, then we would obtain a three dimensional surface area. We would

ideally want this area to be minimized to give an indication of a good controller. This performance measure is illustrated in figure 3.7.

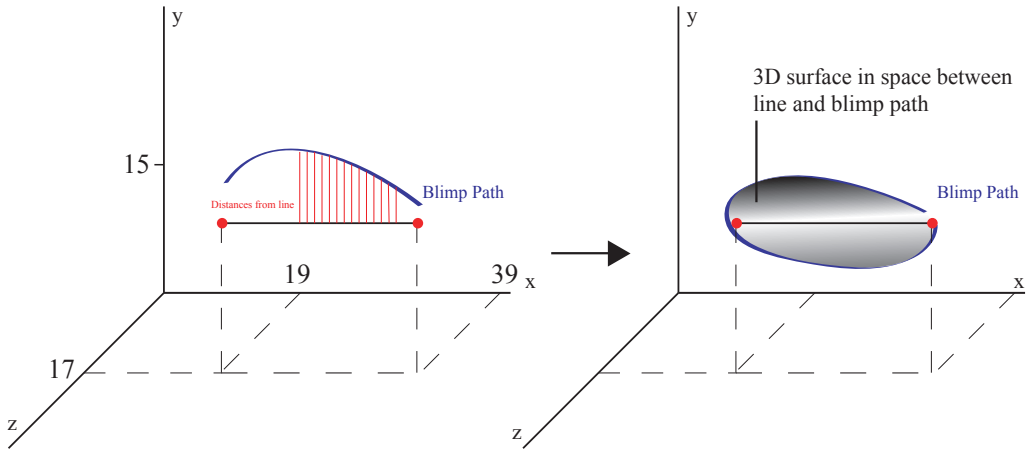

Figure 3.7: Illustration of 3D surface in space performance measure

The previous performance measures were calculated for each controller. They allow us to compare between the different controllers given that the conditions of statistical significance were met. Statistical significance was calculated using the Student's t-test (unless otherwise stated). It is the case that there are so many disturbances acting on the blimp at all times that the test runs may not be a representative sample for comparison. The next section discusses the sources of error that affect the blimp platform.

3.4 Sources of error

This section discusses the problematic issues observed throughout this project. It will highlight the factors that affect the blimp's performance during the test runs.

Initially, the blimp does not have a pressure sensor to calculate the contained helium pressure, so each day of testing the blimp will contain a different amount of helium. The gondola's position also differs slightly at the beginning of each testing day. This has the effect of changing the center of mass for the blimp and could alter the performance.

The blimp has a lot of inertia when moving in a certain direction, thus changing the direction requires some time. It then becomes important that no series of faulty motor commands be sent to the blimp, as this will cause the blimp to stray from its path and it requires more time to set it back onto its course. Wind gusts resulting from elevator

movement, doors opening and closing as well as people moving around in the test area will carry the blimp away from its path. An illustration of the positions of the doors and elevators in the Informatics Forum is shown in figure 3.8.

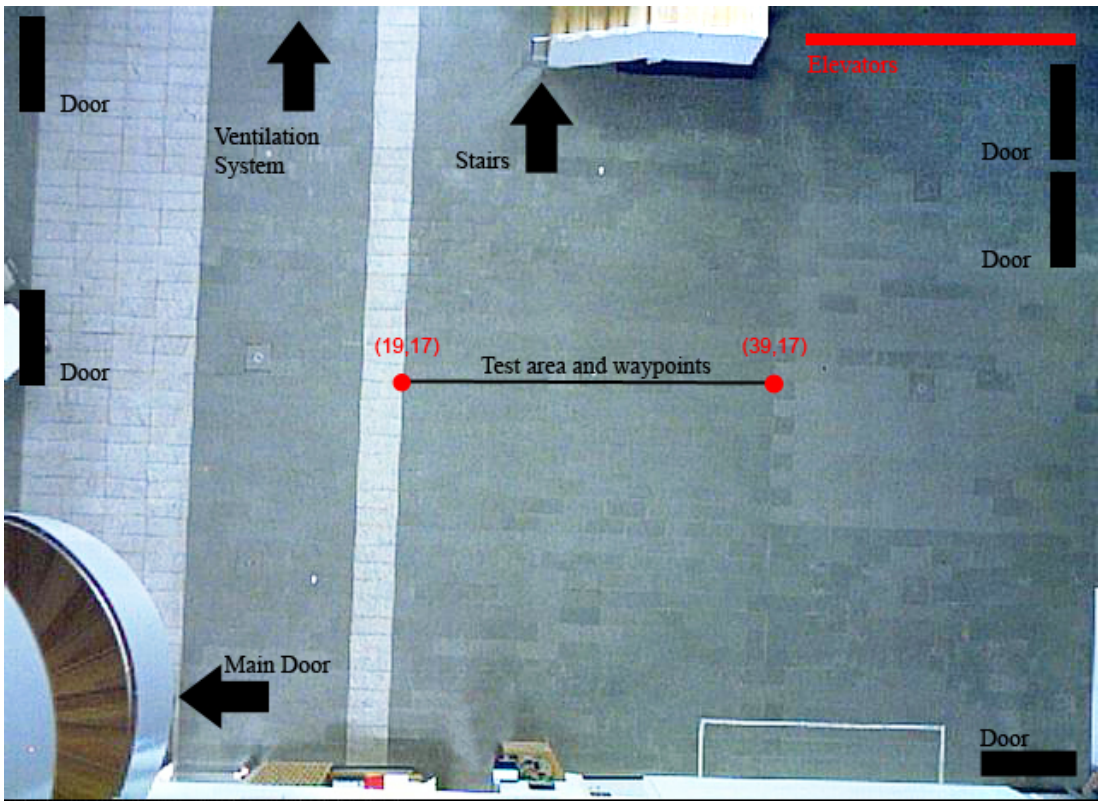

Figure 3.8: Positions of the doors and elevators in the Informatics Forum

The software for the blimp calculates the area of the blimp as seen from a network camera fixed at the ceiling of the Informatics Forum looking down on the testing area. The area is calculated to get a height estimation. Since this is done every image frame, a lot of noise is introduced into the recorded height of the blimp as lighting conditions might change from one captured frame to another. Another software limitation comes in the form of motor command delays as shown in [25]. Other software limitations include how a blimp registers the arrival at a goal. The robotic blimp would register an arrival at the goal if it is in the range of 3 feet of the goal.

The blimp is held by two ropes at each end of its envelope. The amount of rope length given to the blimp affects the weight the blimp is carrying and thus would affect its final height. The amount of rope length can vary throughout the test run, and can affect the results.

Other issues that affect the blimp during its run is change of temperature in the testing area, this will lead to change in pressure and change of the height of the blimp.

Chapter 4

3-Parameter State Based Controller

This chapter discusses the design and implementation of the 3-parameter state based controller applied on the robotic blimp. Overall performance of the robotic blimp under this controller is analyzed. A comparative analysis with and without disturbance is also given.

4.1 Introduction

The robotic blimp has to endure some harsh disturbances in the form of air gusts as well as uncertainties within its parameters. The blimp provides a challenging platform for controller design, and thus becomes a valuable tool for testing different controllers and their robustness. This chapter introduces the first of the controllers that will be addressed in this dissertation. This controller is composed of three stages: height control, 3-parameter state orientation control, and 3-parameter state position control. There is a hierarchy within this controller such that orientation control is triggered once the height control achieves desired height, and the position controller is only triggered when the orientation is stable and facing the target within a tolerance. While the height controller is a very important part of the design, it was fixed for all three controllers investigated in this dissertation. The focus will be on the performance of the controller within the (x,z) plane, namely the 3-parameter state orientation control and 3-parameter states position control.

Further detail on the design of the controller is given in section 4.2, results of the controller performance with and without disturbances is given in sections 4.3.1 and 4.3.2 respectively. Overall performance is given in section 4.3.3. Conclusions and discussion of the results is given in section 4.4.

4.2 Controller design

Sensor inputs in the form of position relative to goal, linear velocity, relative orientation to goal and angular velocity are obtained from the ceiling camera. These sensor inputs are each divided into ranges, if the sensor value at a given instance was within a certain range; it is given a linguistic variable. The controller looks at the current, future and speed states and decides on an output. The outputs are hardcoded, and fixed.

The blimp orientation control algorithm is as follows, once the blimp's absolute orientation is given by the image processing block, the relative orientation is calculated. The angular velocity is then calculated by looking the difference in orientation between the current and the previous image frame. The inputs are then passed to three indicator functions that give linguistic descriptions of the states. The first indicator function gives the present state of the blimp, it can take one of three values: clockwise (cw), anticlockwise (acw) and near, as seen in figure 4.1.

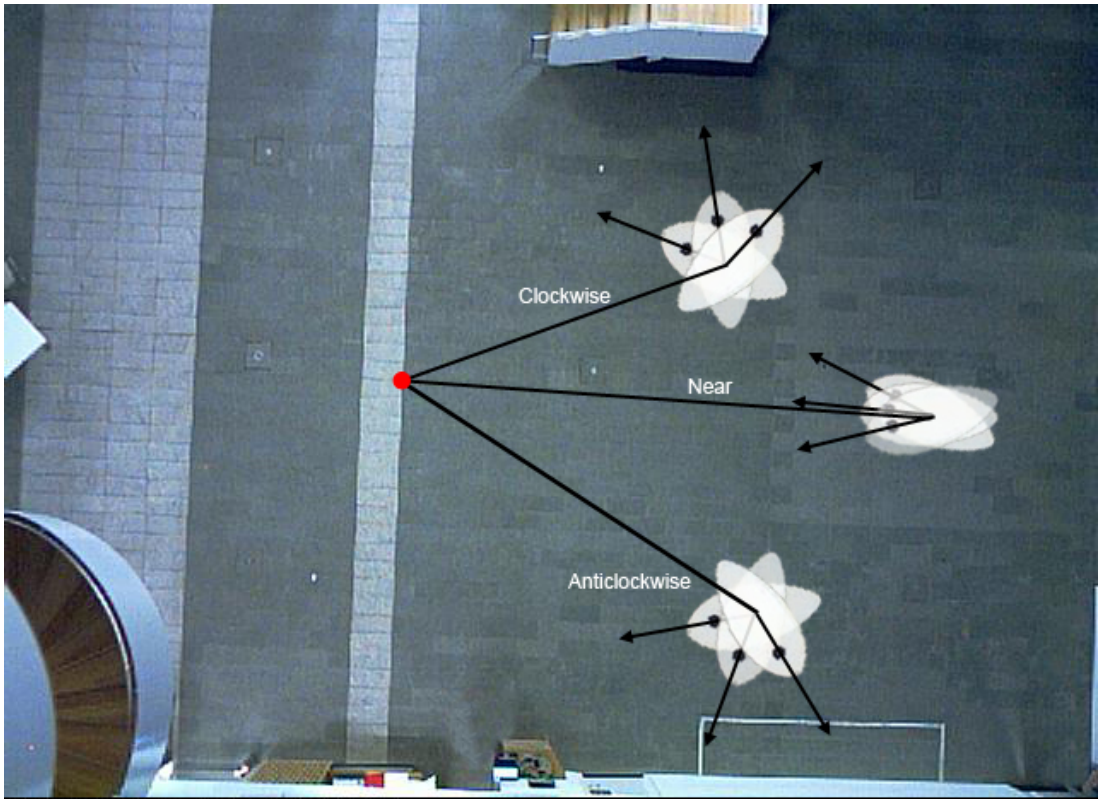

Figure 4.1: Examples of the indicator function's linguistic values for the orientation controller

Clockwise here means that the blimp is within the range of [0, pi] relative to the goal orientation, anticlockwise means that the blimp orientation is in the range of [pi,

2pi] of the goal orientation. The value "near" means that the blimp is within a tolerance of +18 /-18 degrees of the target orientation. The second indicator function gives an indication to the predicted future orientation of the blimp in the next frame, it takes the same values as the present indicator function: clockwise, anticlockwise and near. This is calculated by looking at the current relative orientation to goal and adding it to the angular speed multiplied by the time step. The third and final indicator function is the speed indicator function. It can return a value of fast clockwise if the blimp is rotating clockwise with a speed greater than 0.06-rad/second (3.4-degree/second), or the indicator function can return a value of fast anticlockwise if the blimp is rotating anticlockwise with a speed greater than 0.06-rad/second (3.4-degree/second), or it can return a value of slow if the speed is under 0.06-rad/second in either direction.

At any given instance and as long as the orientation controller is in command, any position of the blimp has three states: its present orientation, its future orientation and the angular velocity. The combination of these three states gives 27 cases the blimp might be in. Depending on the case, a certain action must be taken. There are five hard-coded actions available, they are as follows: accelerate clockwise (25% of maximum motor power), accelerate clockwise fast (37.5% of maximum motor power), do nothing, accelerate anticlockwise (25% of maximum motor power), accelerate anticlockwise fast (37.5% of maximum motor power). The 25% and 37.5% motor commands were obtained through trial and error, to obtain a smooth rotation to goal without overshoot or undershoot.

For example, if the present orientation relative to goal is clockwise and the future orientation relative to goal is also clockwise and the speed direction relative to goal is fast clockwise then we need to move anticlockwise, we decide on how fast anticlockwise we want to go depending on the current angular velocity. If it is less than 0.08-rad/second or 4.6-degrees/second then the controller issues a command to rotate anticlockwise else if it is over 0.08-rad/second then the controller issues a command to rotate fast anticlockwise. This is shown in figure 4.2. There are 26 other cases like the one mentioned in the example above, they take the logical form of: if {Present Orientation} and {Future Orientation} and {Speed} then {Action}. The full list is provided in Appendix A. An important case is when the blimp is in the following case: if { Present Orientation = near} and { Future Orientation = near} and {Speed = slow} then {return "stable"} which then tells the controller that the blimp is pointing towards the goal destination, and the position control should be activated.

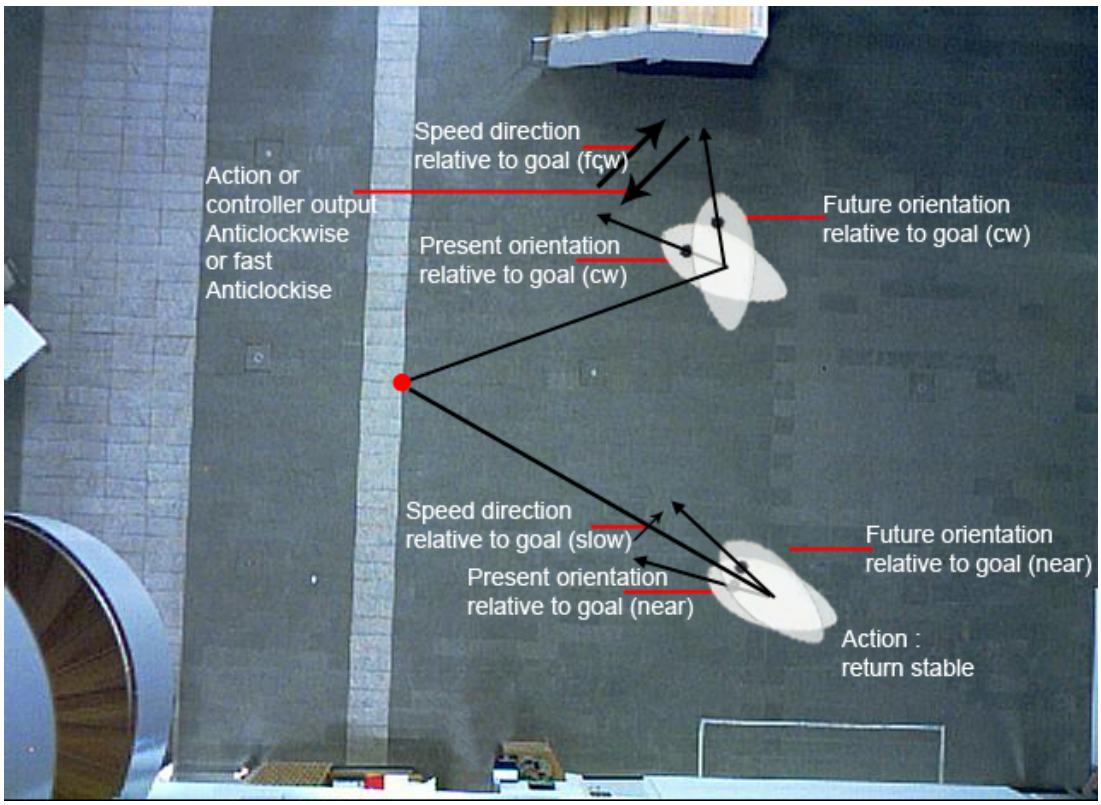

Figure 4.2: 3-parameter orientation controller example

The position controller algorithm has a similar design as the orientation control. However; present position state can take values of: front, beyond and near as shown in figure 4.3. Front is returned if the blimp is facing the goal and is at a distance of 7-feet or more. Beyond is returned if the blimp is not facing the goal and is at a distance of 7-feet or more. Near is given if the blimp is within a 7-feet radius of the goal. The future position state can take the same values of front, beyond and near. Speed can take values of fast forwards if the blimp is moving forward at a speed of over 2-cms/second, fast backwards forwards if the blimp is moving backwards at a speed of over 2-cms/second and slow if the blimp is moving backwards or forwards at a speed less than 2-cms/second. Slow is an indication that the blimp is almost standing still.

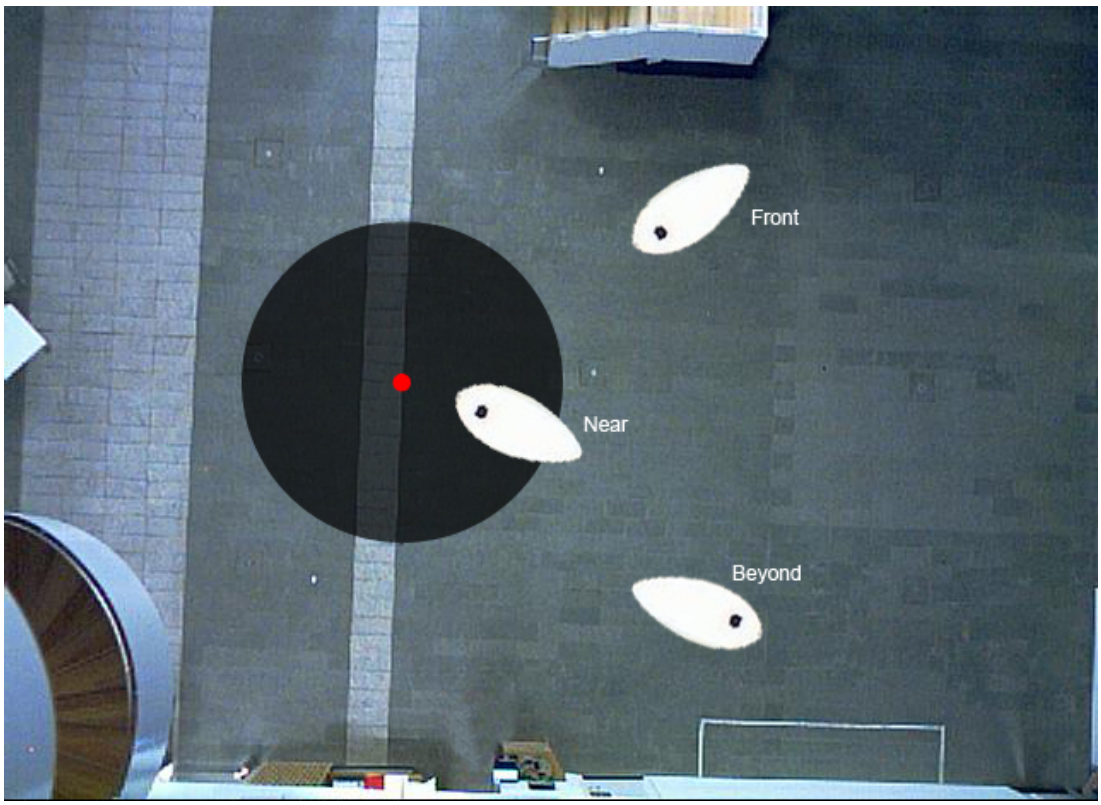

Figure 4.3: Indicator function's linguistic values for the position controller

The outputs for the position controller are as follows: fast forward (30% of maximum motor output), forward (22% of maximum motor output), do nothing, backwards (22% of maximum motor output), and fast backwards (30% of maximum motor output), the do nothing command is 0% of the maximum motor output.

We thus have 3 states, present position, future position and speed. Each state can have three values. The combination of these values makes 27 cases, same as the orientation control. The cases take the form of if {Present Position} and {Future Position} and {Speed} then {Action}. For example, if {Present Position = front} and {Predicted Position = front} and {Speed = fast forward} then {Action = do nothing}. This is because the blimp is heading towards the goal with some speed and we want to rely on its current drift to carry it towards the goal. This is shown in figure 4.4. The full list of the 27 cases is given in Appendix B. It is worth mentioning that some of these cases naturally do not make sense, and could not occur. They are labeled in the appendix and the default controller command is 'do nothing'.

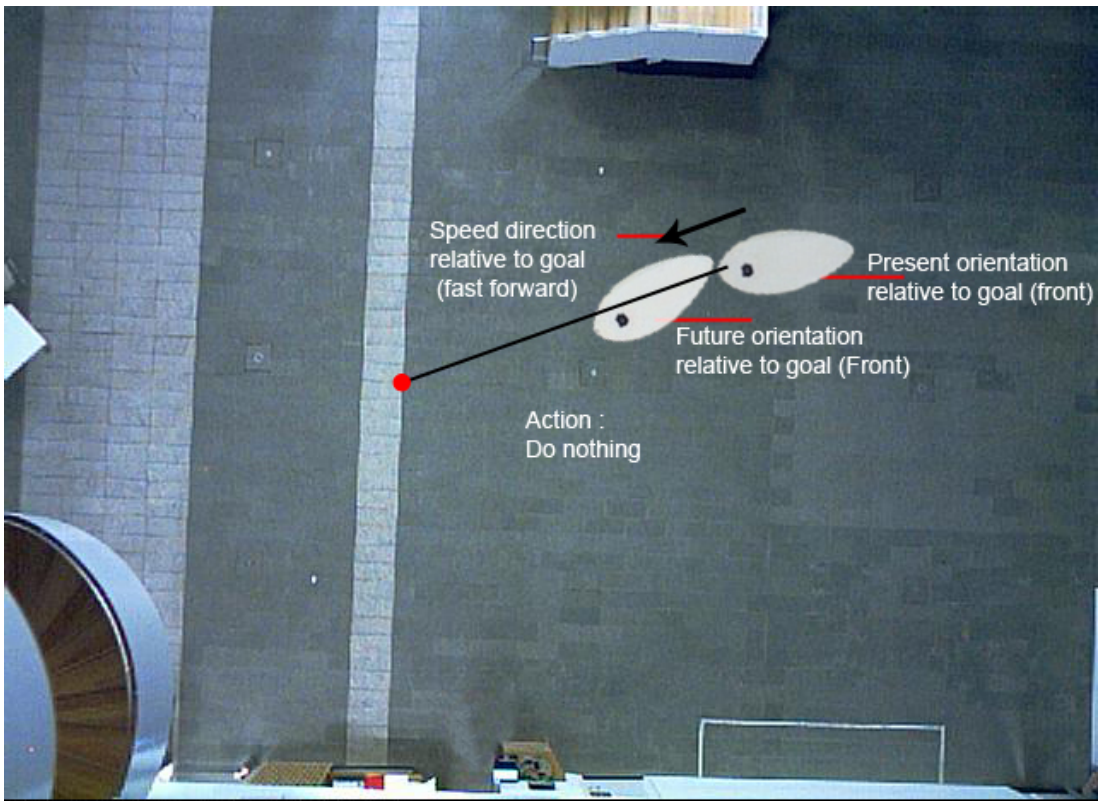

Figure 4.4: 3-parameter position controller example

The next section presents the performance results of this controller design.

4.3 Controller results

This section presents the results of the 3-parameter state based controller. The experiments were run as discussed in section 3.2 and results of the runs will be presented as discussed in sections 3.2 and 3.3. The first subsection below 4.3.1 deals with the results of 10 runs that were recorded in the presence of disturbances in form of constant people movement and/or elevators running. The second subsection 4.3.2 deals with the results of 10 runs that were recorded without such disturbances. The third subsection 4.3.3 deals with the combined results of the combined 20 runs (with and without disturbances).

4.3.1 Controller analysis with disturbances

This section presents the results of ten runs where disturbances were present when applying the 3-parameter state based controller. These disturbances can be in the form

of people moving across the test area, people opening and closing doors of the Forum and using the elevators, all of which induce some kind of wind gusts that may affect the performance of the robotic blimp.

Figure 4.5 shows the average of the x position vs. motor commands of the blimp under disturbances, the green line presents the average across the ten runs, and the black lines show one standard deviation from the mean. Figure 4.6 shows the average of the z position vs. motor commands of the blimp under disturbances, the red line presents the average across the ten runs, and the pink lines show one standard deviation from the mean. Figure 4.7 shows the average of the y position (height) vs. motor commands of the blimp under disturbances, the blue line presents the average across the ten runs, and the cyan lines show one standard deviation from the mean. Figure 4.8 finally shows the top view of the average lap taken by the blimp between the two points (39, 17) and (19, 17).

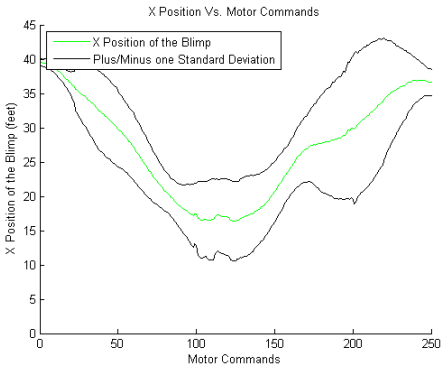

Figure 4.5: X position vs. Motor commands with disturbances

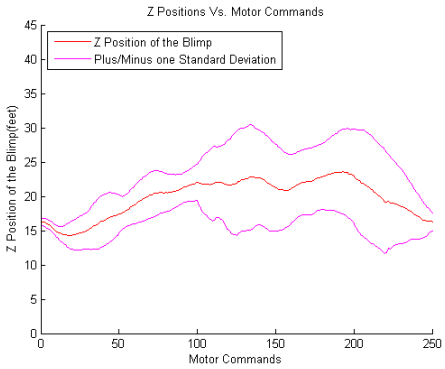

Figure 4.6: Z position vs. Motor commands with disturbances

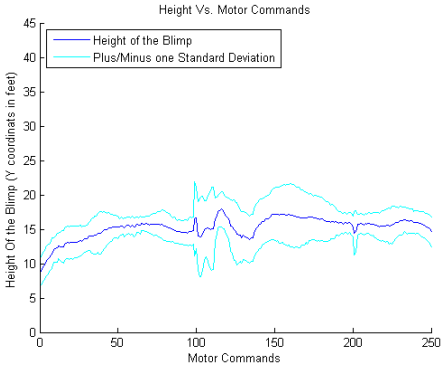

Figure 4.7: Height vs. Motor commands with disturbances

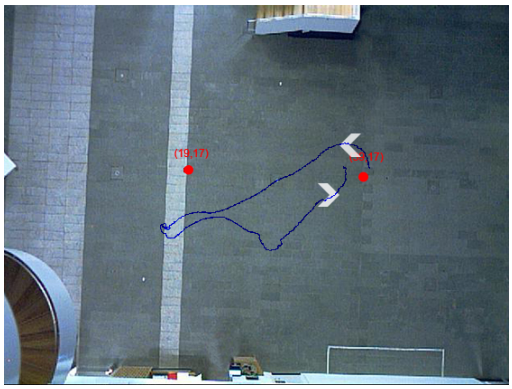

Figure 4.8: Z position vs. X position with disturbances

Table 4.1 gives the performance measures calculated of the average run as well as the standard deviations from those averages. The area between the standard deviations is also calculated. The effects of the disturbances can be viewed by the large standard deviations, and the instability of the blimp's height as seen in figure 4.7. The effect of the disturbances can also be seen in figure 4.8 where the blimp was always drifting towards the main door due to the air currents. The blimp was always overshooting the first waypoint (19, 17) and was not maintaining a forward heading towards it. The blimp was also not successful at maintaining a straight path on the way back from (19, 17) to (39, 17).

Value	Average	Standard deviation
Time (minutes)	3.7182	1.2073
Average x (feet)	27.2630	7.3746
Average z (feet)	19.8259	2.7527
Average y (feet)	15.2351	1.4731
Area between x standard deviations	2596	
Area between z standard deviations	2235	
Area between y standard deviations	1289	
Area in 3D	3.0503e+003	856.0986

Table 4.1: 3-parameter state based controller with disturbances results

4.3.2 Controller analysis without disturbances

This section presents the results of ten runs where disturbances were not present when applying the 3-parameter state based controller. Disturbances in the form of people moving around usually stops after 7:30 pm, and this is usually when these results were obtained.

Figure 4.9 shows the average of the x position vs. motor commands of the blimp under no disturbances, the green line presents the average across the ten runs, and the black lines show one standard deviation from the mean. Figure 4.10 shows the average of the z position vs. motor commands of the blimp under no disturbances, the red line presents the average across the ten runs, and the pink lines show one standard deviation from the mean. Figure 4.11 shows the average of the y position (height) vs. motor commands of the blimp under no disturbances, the blue line presents the average across the ten runs, and the cyan lines show one standard deviation from the mean. Figure 4.12 finally shows the top view of the average lap taken by the blimp between the two points (39, 17) and (19, 17).

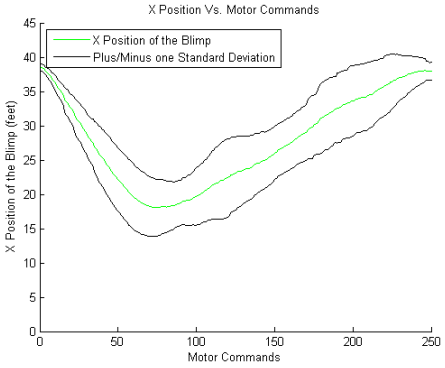

Figure 4.9: X position vs. Motor commands without disturbances

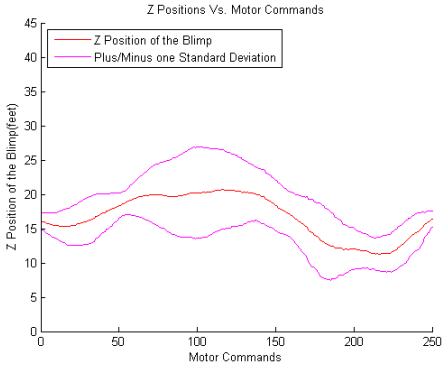

Figure 4.10: Z position vs. Motor commands without disturbances

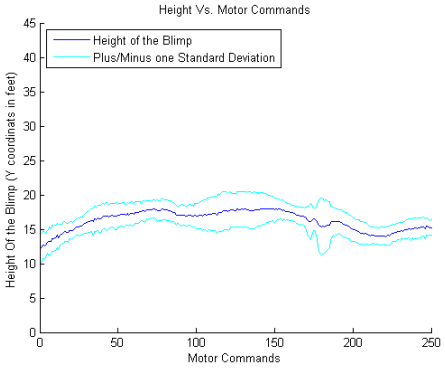

Figure 4.11: Height vs. Motor commands without disturbances

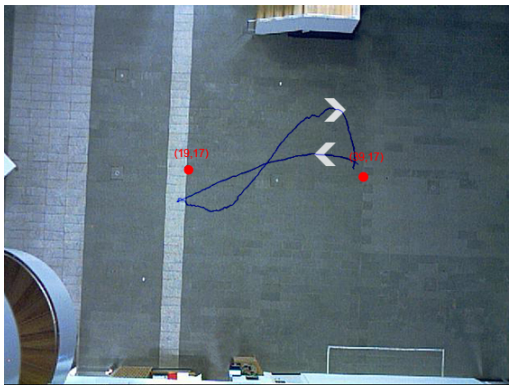

Figure 4.12: Z position vs. X position without disturbances

Table 4.2 gives the performance measures calculated of the average run as well as the standard deviations from those averages. The area between the standard deviations is also calculated. The effects of the absence of disturbances can be viewed by the lower standard deviations in general, and the low standard deviations when looking at figure 4.11. The average run is smoother and heads towards goal better in its path from (39, 17) to (19, 17) with some overshoot. The average inbound path from (19, 17) to (39, 17) showed signs of overshooting the desired orientation. The blimp recovered from the overshoot by the end of the journey and arrived at the goal at waypoint (39, 17).

Value	Average	Standard deviation
Time (minutes)	3.3893	0.6073
Average x (feet)	27.7588	6.6485
Average z (feet)	16.6533	3.1534
Average y (feet)	16.2591	1.4175
Area between x standard deviations	1962	
Area between z standard deviations	1863	
Area between y standard deviations	895	
Area in 3D	2.3779e+003	1.1376e+003

Table 4.2: 3-parameter state based controller without disturbances results

4.3.3 Overall controller analysis

This section presents the results of all twenty runs with the application of the 3-parameter state based controller. They are a mixture of runs with and without disturbances; these results will be used to compare the three different controllers under study in chapter 7.

Figure 4.13 shows the average of the x position vs. motor commands of the blimp, the green line presents the average across the twenty runs, and the black lines show one standard deviation from the mean. Figure 4.14 shows the average of the z position vs. motor commands of the blimp, the red line presents the average across the twenty runs, and the pink lines show one standard deviation from the mean. Figure 4.15 shows the average of the y position (height) vs. motor commands of the blimp, the blue line presents the average across the twenty runs, and the cyan lines show one standard deviation from the mean. Figure 4.16 finally shows the top view of the average lap taken by the blimp between the two points (39, 17) and (19, 17).

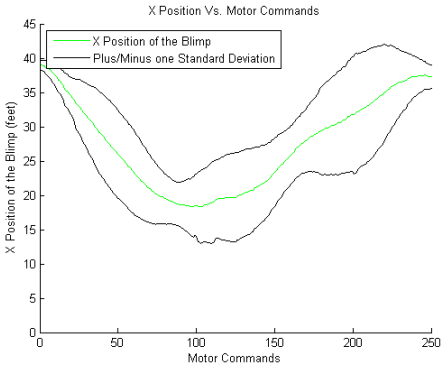

Figure 4.13: X position vs. Motor commands integrated with and without disturbances

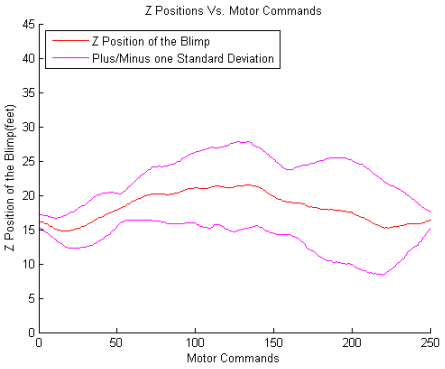

Figure 4.14: Z position vs. Motor commands integrated with and without disturbances

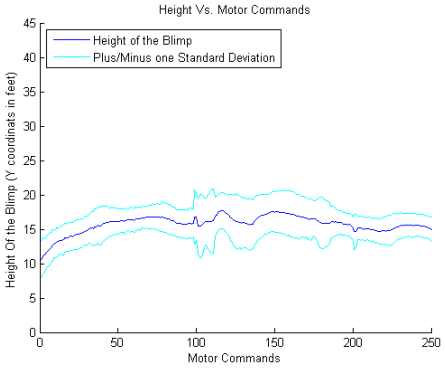

Figure 4.15: Height vs. Motor commands integrated with and without disturbances

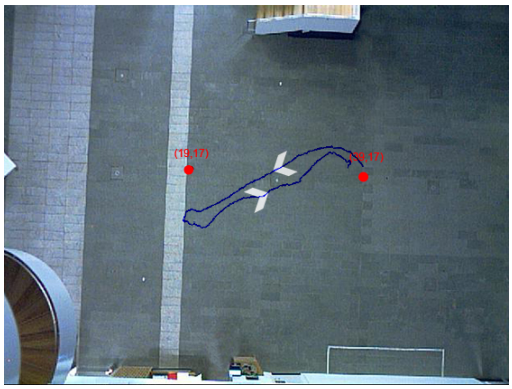

Figure 4.16: Z position vs. X position integrated with and without disturbances

Table 4.3 gives the performance measures calculated of the average run as well as the standard deviations (std.) from those averages. The area between the standard deviations is also calculated.

Value	Average	Standard deviation
Time (minutes)	3.5538	0.9453
Average x (feet)	27.5109	6.6638
Average z (feet)	18.2396	2.1913
Average y (feet)	15.7471	1.2264
Area between x standard deviations	2573	
Area between z standard deviations	2397	
Area between y standard deviations	1221	
Area in 3D	2.7141e+003	1.0388e+003

Table 4.3: 3-parameter state based controller results

Overall the controller has been able to achieve the task in each run, however as seen from figure 4.16, but it is not able to head directly to the goal and is affected by different disturbances. The blimp also fails on average to pass cleanly through point (19, 17) and overshoots it. Further analysis of the overall performance is discussed in the following section.

4.4 Conclusion and discussion

This section discusses the overall performance of the 3-parameter state based controller. It also provides a comparative analysis of the controller with and without disturbances.

The following statistical significance work in this section was done using t-test for unpaired samples one-tailed hypothesis test; findings are summarized and discussed in this section. Table 4.4 summarizes the numerical results for the controller under the different conditions.

Comparison metric	Under disturbances	Under no disturbances	Overall performance
Area in 3D	3.0503e+003	2.3779e+003	2.7141e+003
Average x	27.2630	27.7588	27.5109
Average z	19.8259	16.6533	18.2396
Average y	15.2351	16.2591	15.7471
Area between x std.	2596	1962	2573
Area between z std.	2235	1863	2397
Area between y std.	1289	895	1221

Table 4.4: Comparative results for 3-parameter state based controller.

The average area in 3D for a state based controller under disturbances is higher than that for the same controller without disturbances ($t = 1.49334$, degrees of freedom (DF)=18.0, $p<0.08$).

The x position's standard deviations for the 10 runs for the state based controller under disturbances were higher than that for the same controller without disturbances ($t = 7.43457$, DF=364.35 , $p<0.05$ using Satterthwaite's approximate t-test for unpaired samples). The Fisher-Snedecor f-test for equality of variances showed that the variances are different ($F = 4.07147$, Dfn = Dfd =249 , $p<0.05$). The degrees of freedom in the Satterthwaite's t-test are approximated using Welch-Satterthwaite equation (rather than sample size), that is why we get fractional results for DF.

The z position's standard deviations for the state based controller under disturbances were higher than that for the same controller without disturbances ($t = 4.71235$, DF=441.25 , $p<0.05$ using Satterthwaite's approximate t-test for unpaired samples). The Fisher-Snedecor f-test for equality of variances showed that the variances are different ($F = 2.11823$, Dfn = Dfd =249 , $p<0.05$).

The y position's standard deviations for the state based controller under disturbances were higher than that for the same controller without disturbances ($t = 10.05757$, $DF = 377.0237$, $p < 0.05$ using Satterthwaite's approximate t-test for unpaired samples). The Fisher-Snedecor f-test for equality of variances showed that the variances are different ($F = 3.61314$, $Dfn = Dfd = 249$, $p < 0.05$).

Results show that the controller has preformed worst under disturbances as expected. The area made in 3D space was 3050.3-feet^2 under disturbances with comparison to the 2377.9-feet^2 produced under no disturbances indicating a journey that is longer and with more deviation from the optimum path. The areas between standard deviations were always tighter and smaller in the case of no disturbances, indicating a change in performance in the existence of environmental disturbances.

However; the controller managed to recover from these disturbances and proved to be robust to environmental effects. This leads to the question of what more can be done to achieve a better performance and avoid overshooting the waypoints? The next chapter addresses this issue and introduces an addition to the controller design by looking at the previous step the blimp was in rather than just the present and the future. This would enhance the controller's understanding of its current location and removes false actions that could shift the momentum of the blimp away from the correct course.

Chapter 5

4-Parameter State Based Controller

This chapter presents the discussion and the design of the 4-parameter state based controller. Introduction and motivation are given in section 5.1. Section 5.2 presents controller design and examples of the algorithmic logic. Section 5.3 shows the results of the controller under disturbances and without disturbances and shows the robotic blimp's overall performance under the 4-parameter state based controller. A discussion of the results is given in section 5.4.

5.1 Introduction

The state based controlled discussed in chapter 4 performed well. It completed its laps and worked against environmental disturbances. However; we would like to further enhance the performance of the robotic blimp in such a way to minimize deviation from the straight path connecting the two way points, and increase the robustness of the controller to all kind of disturbances. A problem that was noticed during taking the test runs was the issue of shift of momentum. The blimp has large inertia, it takes time to get it to move in a certain direction, and it takes longer time to stop that movement and drive it in another direction. If a series of wrong actions were triggered, the blimp would be forced to move to a wrong direction, increase the error and would waste time until recovery.

In this chapter an addition is proposed to the previous controller. This addition comes in the form of memory. Instead of just looking at the present state, predicted future state and speed of the blimp, the blimp also uses the previous state that it was in. This would disambiguate any confusion of where its current state is and would remedy any problems of faulty momentum shifts. It will also allow the controller designer to

get a better understanding where the blimp is and what the blimp motor commands should be at that point.

More information on the design of the controller as well as examples are given in the next section.

5.2 Controller design

This section presents a discussion on the design process of the 4-parameter state based controller.

The controller design is exactly the same as that discussed in section 4.2 but in addition to the 3 states considered: present state, future state and speed, we have an addition of previous state. The conditional if statement becomes if {Previous state} and {Present state} and {Future state} and {Speed} then {Action}.

This is applied to both the orientation controller and the position controller. The hierarchy is still the same such that if the height controller achieves the desired height, only then would the orientation controller would be triggered and the position controller would only be triggered when the orientation controller is stable and is heading to the goal.

To understand the importance of this addition a discussion and some examples are given in the following paragraph. As stated earlier the state based controller with no history relies on the present state and predicted future state to make sense of its current state. However; if they are the same, and have the same linguistic value, then it would be useful to obtain the previous state to resolve ambiguity. For example, and as shown in figure 5.1 if the orientation controller's present state is 'clockwise' and the future state is 'clockwise' then by knowing the previous state 'clockwise' we can assume that the blimp has lost momentum and is stuck and needs extra power to get it to the correct orientation. If the previous state was 'near' then we can understand that it has left the desired orientation and heading the wrong way, extra power is needed to reverse the rotation and get it back to head towards the goal. If the previous state was 'anticlockwise' we can understand that the blimp has crossed the 180-degree point and is heading towards goal and may need power to achieve the correct heading. This example becomes more important if the speed has the linguistic variable of 'slow' which does not give a direction to the angular velocity of the blimp in comparison to 'fast clockwise' and 'fast anticlockwise'.

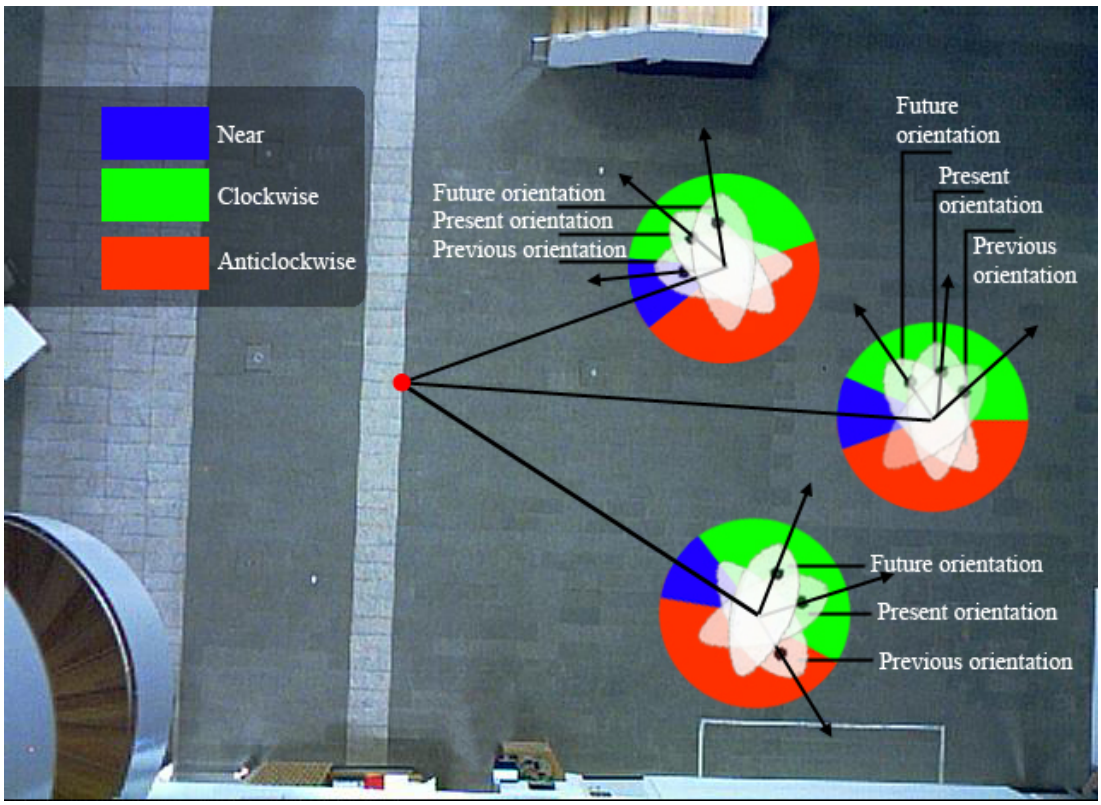

Figure 5.1: 4-parameter orientation controller example

The same argument can be made for the position controller where the previous state would give us a better indication of how far the blimp is from its goal. It would also allow us to resolve any faulty cases especially when the speed state is 'slow' and does not give the designer an indication of blimp movement, the previous state allows us to understand the direction of movement in that case. For example, if {Previous state = front} and {Present state = near} and {Predicted state = near} and {Speed = slow} then {Action = forward}, as the boundary of the front-near state has just been passed and some power is required to drive it to goal when its speed is slow. However; if {Previous state = near} and {Present state = near} and {Predicted state = near} and {Speed = slow} then {Action = do nothing}, as it is on its way drifting towards goal. But if {Previous state = beyond} and {Present state = near} and {Predicted state = near} and {Speed = slow} then {Action = reverse} as the blimp had just crossed the border from the beyond state and we want it to move backwards. This example illustrates the importance of having a previous state, without which a wrong motor command may be issued sending the blimp in the wrong direction. A full list of the actions of the orientation controller is given in Appendix C. Actions of the position controller are given in Appendix D.

5.3 Results

This section presents the results of the 4-parameter state based controller. The experiments were run as discussed in section 3.2 and results of the runs will be given as discussed in sections 3.2 and 3.3. Subsection 5.3.1 deals with the results of 10 runs that were recorded in the existence of disturbances in form of constant people movement and/or elevators running. Subsection 5.3.2 deals with the results of 10 runs that were recorded without such disturbances. The third subsection deals with the results of 20 runs that were recorded without noting the existence of disturbances or not.

5.3.1 4-parameter state based controller under disturbances

This section presents the results of ten runs where disturbances were present when applying the 4-parameter state based controller. These disturbances can be in the form of people moving across the test area, people opening and closing doors of the Forum and using the elevators, all of which induce some kind of wind gusts that may affect the performance of the robotic blimp. These runs were recorded between 5pm – 7:30pm at the Informatics Forum when people are usually leaving the building. Figure 5.2 shows the average of the x position vs. motor commands of the blimp under disturbances, the green line presents the average across the ten runs, and the black lines show one standard deviation from the mean. Figure 5.3 shows the average of the z position vs. motor commands of the blimp under disturbances, the red line presents the average across the ten runs, and the pink lines show one standard deviation from the mean. Figure 5.4 shows the average of the y position (height) vs. motor commands of the blimp under disturbances, the blue line presents the average across the ten runs, and the cyan lines show one standard deviation from the mean. Figure 5.5 finally shows the top view of the average lap taken by the blimp between the two points (39, 17) and (19, 17).

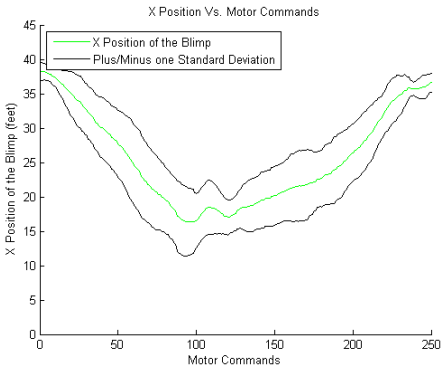

Figure 5.2: X position vs. Motor commands with disturbances

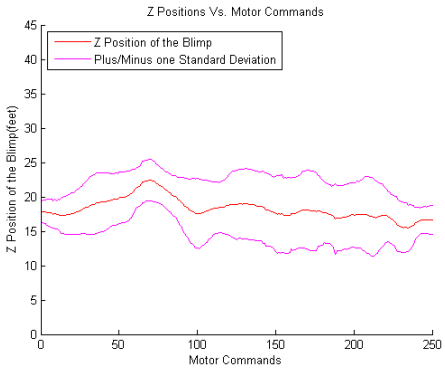

Figure 5.3: Z position vs. Motor commands with disturbances

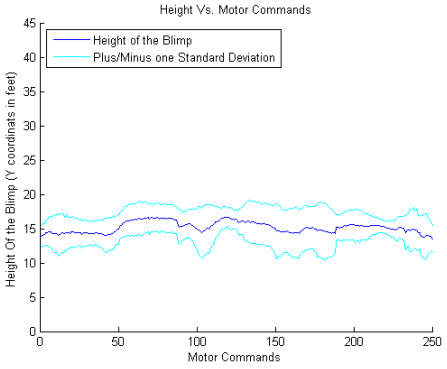

Figure 5.4: Height vs. Motor commands with disturbances

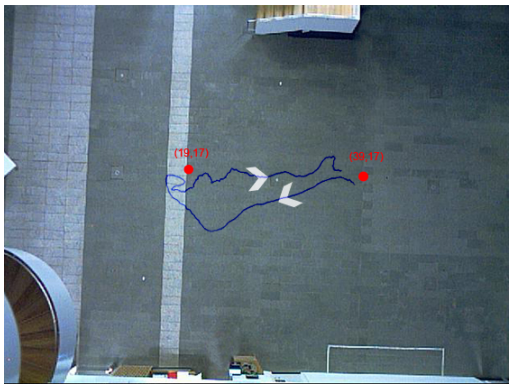

Figure 5.5: Z position vs. X position with disturbances

Table 5.1 gives the performance measures calculated for the average run as well as the standard deviations from those averages. The area between the standard deviations is also calculated. The effects of the disturbances can be viewed by the large standard deviations, and the instability of the blimp's height as seen in figure 5.4; where the average is always changing. The effect of disturbances can also be seen in figure 5.5 where the blimp took a large curve in its inward approach to waypoint (19, 17), but it always recovered and managed to achieve the first waypoint and turn towards the second waypoint. The blimp was always changing direction on its way back to the second waypoint.

Value	Average	Standard deviation
Time (minutes)	4.4620	1.5324
Average x (feet)	25.6323	6.8630
Average z (feet)	18.2686	1.5151
Average y (feet)	15.1725	0.7982
Area between x standard deviations	1931	
Area between z standard deviations	2027	
Area between y standard deviations	1209	
Area in 3D	2.4271e+003	1.3677e+003

Table 5.1: 4-parameter state based controller under disturbances results

5.3.2 4-parameter state based controller under no disturbances

This section presents the results of ten runs where disturbances were not present when applying the 4-parameter state based controller. Disturbances in the form of people moving around usually stop after 7:30 pm in the Forum, and this is usually when these results were obtained.

Figure 5.6 shows the average of the x position vs. motor commands of the blimp under no disturbances, the green line presents the average across the ten runs, and the black lines show one standard deviation from the mean. Figure 5.7 shows the average of the z position vs. motor commands of the blimp under no disturbances, the red line presents the average across the ten runs, and the pink lines show one standard deviation from the mean. Figure 5.8 shows the average of the y position (height) vs. motor commands of the blimp under no disturbances, the blue line presents the average across the ten runs, and the cyan lines show one standard deviation from the mean. Figure 5.9 finally shows the top view of the average lap taken by the blimp between the two points (39, 17) and (19, 17).

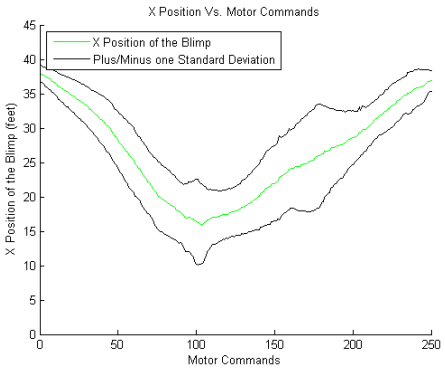

Figure 5.6: X position vs. Motor commands with no disturbances

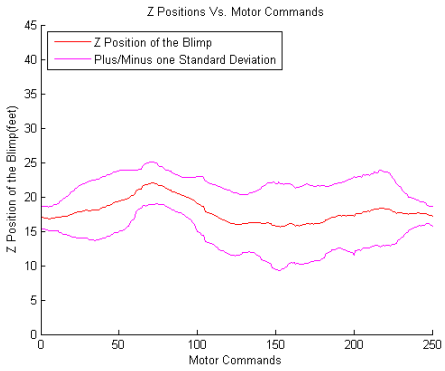

Figure 5.7: Z position vs. Motor commands with no disturbances

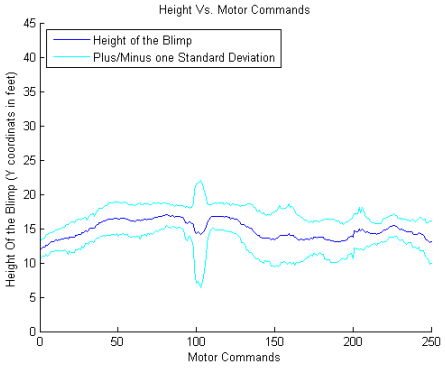

Figure 5.8: Height vs. Motor commands with no disturbances

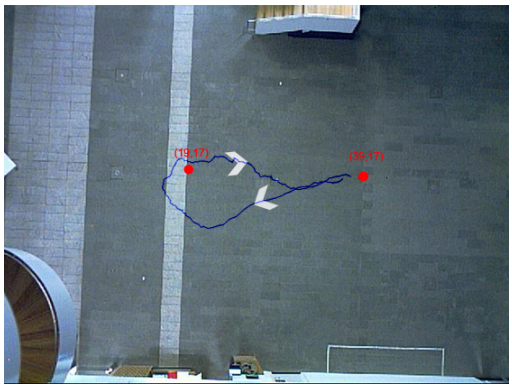

Figure 5.9: Z position vs. X position with no disturbances

Table 5.2 gives the performance measures calculated for the average run as well as the standard deviations from those averages. The area between the standard deviations is also calculated. The effects of the absence of disturbances can be viewed by the low standard deviations in general and the absence of a large number of valleys and mountains in figures 5.6 through 5.8. The average run is smooth and heads towards goal, makes the 180-degree turn and goes to the second way point.

Value	Average	Standard deviation
Time (minutes)	4.0812	1.0726
Average x (feet)	26.4030	6.6739
Average z (feet)	17.8584	1.6785
Average y (feet)	14.8261	1.2922
Area between x standard deviations	2067	
Area between z standard deviations	2058	
Area between y standard deviations	1253	
Area in 3D	2.1739e+003	1.2660e+003

Table 5.2: 4-parameter state based controller under no disturbances

5.3.3 Overall 4-parameter state based controller results

This section presents the results of all twenty runs with the application of the 4-parameter state based controller. They are a mixture of runs with and without disturbances; these results will be used to compare the three different controllers in chapter 7.

Figure 5.10 shows the average of the x position vs. motor commands of the blimp, the green line presents the average across the twenty runs, and the black lines show one standard deviation from the mean. Figure 5.11 shows the average of the z position vs. motor commands of the blimp, the red line presents the average across the twenty runs, and the pink lines show one standard deviation from the mean. Figure 5.12 shows the average of the y position (height) vs. motor commands of the blimp, the blue line presents the average across the twenty runs, and the cyan lines show one standard deviation from the mean. Figure 5.13 finally shows the top view of the average lap taken by the blimp between the two points (39, 17) and (19, 17).

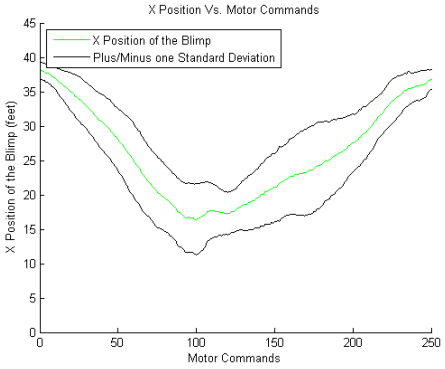

Figure 5.10: X position vs. Motor commands integrated with and without disturbances

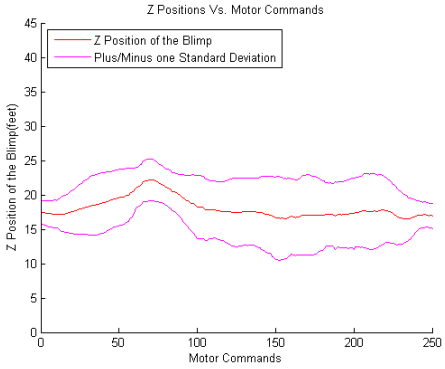

Figure 5.11: Z position vs. Motor commands integrated with and without disturbances

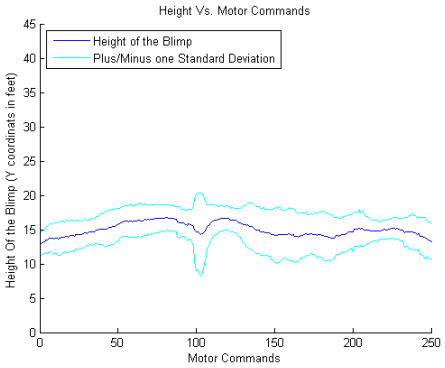

Figure 5.12: Height vs. Motor commands integrated with and without disturbances

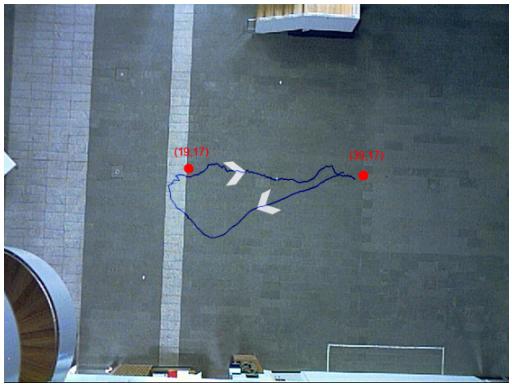

Figure 5.13: Z position vs. X position integrated with and without disturbances

Table 5.3 gives the performance measures calculated for the average run as well as the standard deviations from those averages. The area between the standard deviations is also calculated. Overall the controller has performed well as can be seen in the plots of figures 5.10 through 5.12. The average run in 5.13 is smooth and the blimp heads towards goal in a wide curve, makes the 180-degree turn and goes to the second way point. The blimp moves in a wide curve towards the first way point (19, 17) indicating that the orientation controller is not as aggressive as it should be and may need more tuning to achieve a more direct path to the first way point. There was little overshoot after arriving at the first waypoint and the journey towards the second waypoint (39,

17) was more direct. The dip in figure 5.12 (at motor command = 100) has been noticed in experimentation and it happens when the 180-degree turn is made. When the robotic blimp turns, it spirals down losing height and is more susceptible to disturbances that can pull it down even further.

Value	Average	Standard deviation
Time (minutes)	4.2716	1.3021
Average x (feet)	26.0177	6.7371
Average z (feet)	18.0635	1.4619
Average y (feet)	14.9993	0.9519
Area between x standard deviations	2046	
Area between z standard deviations	2074	
Area between y standard deviations	1273	
Area in 3D	2.3005e+003	1.2893e+003

Table 5.3: Overall 4-parameter state based controller results

5.4 Conclusion and discussion

This section discusses the overall performance of the 4-parameter state based controller. It also provides a comparative analysis of the controller with and without disturbances.The following statistical significance work was done using Student's t-test for unpaired samples one-tailed hypothesis test; findings are summarized and discussed in this section. Table 5.4 summarizes the numerical results for the controller under the different conditions.

Comparison metric	Under disturbances	Under no disturbances	Overall performance
Area in 3D	2.4271e+003	2.1739e+003	2.3005e+003
Average x	25.6323	26.4030	26.0177
Average z	18.2686	17.8584	18.0635
Average y	15.1725	14.8261	14.9993
Area between x std.	1931	2067	2046
Area between z std.	2027	2058	2074
Area between y std.	1209	1253	1273

Table 5.4: Comparative results for 4-parameter state based controller.

Results have shown no significant difference between the controller under disturbances or under no disturbances, indicating that the controller works well in each case and is robust to environmental disturbances. The controller has performed very well; taking a long turn towards the first waypoint, and heading towards the second waypoint directly to complete the run. This effect of taking a long turn towards one of the waypoints might arise from the fact that the controller has single hardcoded outputs, while in fact it might need larger outputs with larger errors, and smaller outputs with smaller errors. Additional work on making the turn more aggressive and making output values relative to errors could solve this problem. This is introduced in the next chapter; chapter six proposes a solution to this problem by using fuzzy logic control to calculate the outputs from a continuous range of outputs. Fuzzy logic control is also investigated to increase the blimp's robustness to environmental disturbances and noise.

Chapter 6

Fuzzy Logic Controller

This chapter introduces the design parameters of the fuzzy logic controller applied to the robotic blimp in section 6.2. The performance of the controllers is discussed in section 6.3. An introduction to fuzzy logic control and motivation is given in section 6.1.

6.1 Introduction

As stated earlier, the blimp is a tough platform to control. It suffers from a lot of disturbances and uncertainties. These disturbances can deviate it from its course and affect its overall performance. The previous two chapters discussed controllers that had hardcoded controller outputs, and inputs that can only take one linguistic variable at a time, e.g. 'cw', 'acw' , 'near' for orientation control. This is very similar to a singleton fuzzy logic control; a special kind of fuzzy logic control that produces discrete outputs. This chapter aims to study the effect of applying a fuzzy logic controller to the robotic blimp where its inputs can be part of more than one linguistic term, and its motor commands can take a wider range of outputs. An introduction to fuzzy logic control is given in the next section through an example.

Fuzzy logic control has also been chosen because of its robustness to noise, model uncertainties and environmental disturbances. Fuzzy logic controllers are also easy to design, they mimic human understanding of the control problem through the utilization of a set of if-then rules called a rule base. They also are easy to tune manually and automatically through adaptive techniques like neural networks.

The next section discusses the steps used to design the fuzzy logic controller.

6.2 Fuzzy logic controller design

This section presents the design parameters of the fuzzy logic controller applied to the robotic blimp. A small introduction to fuzzy logic controller design is also given. An important source for design of fuzzy logic controllers can be found in [13].

Fuzzy logic control can be defined as "control with sentences rather than equations" [13]; the controller uses a set of if-then rules (rule base) rather than a transfer function or a set of dynamic equations. The structure of a fuzzy logic controller involves: fuzzification, rules base and inference engine, and finally defuzzification. The controller designer may choose to use preprocessing and postprocessing techniques. These terminologies will be discussed alongside the design process discussion in the following subsections. The design of the position controller will be discussed in section 6.2.1, the orientation controller will be discussed in section 6.2.2. It is important to note that the position controller will not be activated unless the orientation of the robotic blimp is within +18/-18 degrees range relative to the goal, and its speed is under 0.06 rad/second.

6.2.1 Fuzzy position controller

The structure of the fuzzy orientation controller is explained in this section.

Fuzzification refers to the process of transforming crisp, numerical sensor input into percentage degrees of membership of fuzzy sets. As an example, the present position controller discussed in chapter 4 gave 100% membership to one of three sets: forward, beyond or near. If fuzzy logic control was applied to those sets, then an input (distance from goal) can be a member of many sets, for example 60% forward and 40% near (memberships should add up to 100%). Fuzzification is done through looking at the membership functions assigned to that specific variable. Membership functions can take many forms: Gaussian, Triangular, and Trapezoidal are some examples. We use triangular memberships in our design. An example of triangular membership functions is given in figure 6.1 where the input value of the distance of the robotic blimp from its current goal can take any of 5 triangular membership functions. The five fuzzy sets are: small, medium small, medium, medium large and large. Figure 6.2 shows the membership functions for the speed of the robotic blimp, the three fuzzy sets are: negative, zero and positive. Figure 6.3 shows the membership function for the output that will be fed to the blimp's motors, the fuzzy sets are: small, medium and large. The blimp's motor output can take a range of [0,128], where 0 is 0% of maximum

output and 128 is 100% of maximum output. The output range selected in figure 6.3 was based on trial and error and such that no overshoot would occur.

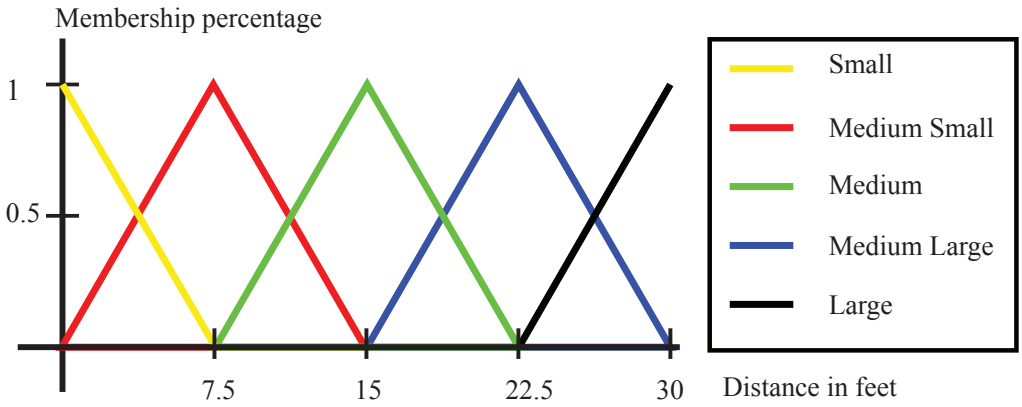

Figure 6.1: Fuzzy position controller distance input membership functions

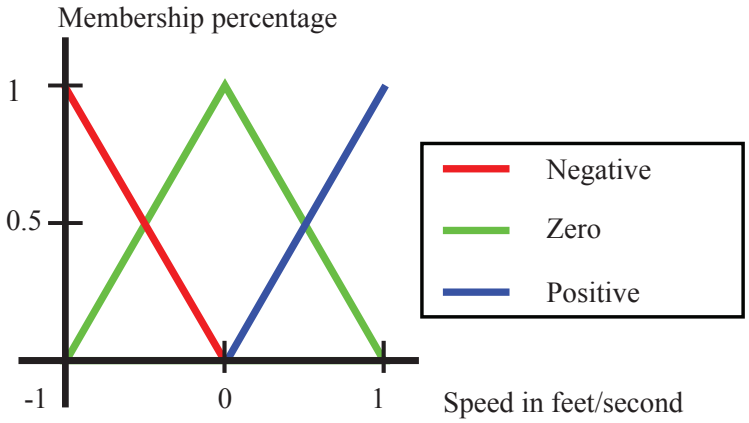

Figure 6.2: Fuzzy position controller speed input membership functions

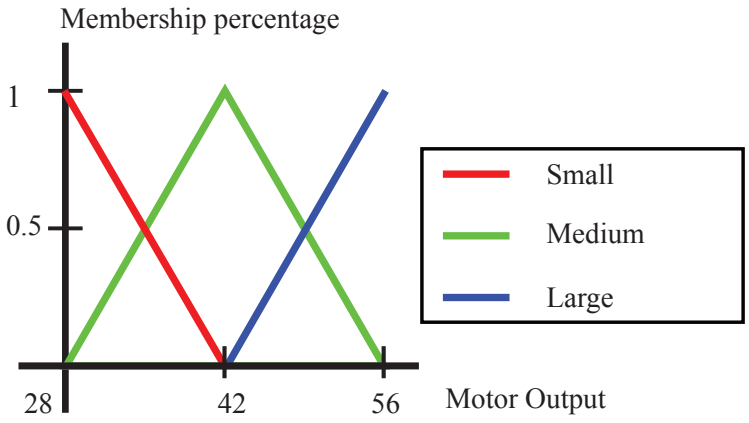

Figure 6.3: Fuzzy position controller output membership functions

An example of how fuzzification works is given in figure 6.4 and figure 6.5. In this example we want to fuzzify the crisp input of 10 feet, shown in figure 6.4, this value is the blimp's distance from the goal. We also want to fuzzify the speed value of zero; this is shown in figure 6.5.

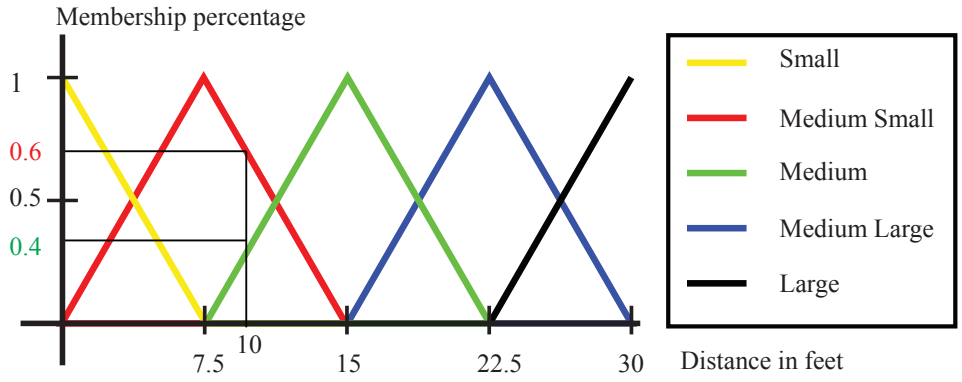

Figure 6.4: Distance input fuzzification example

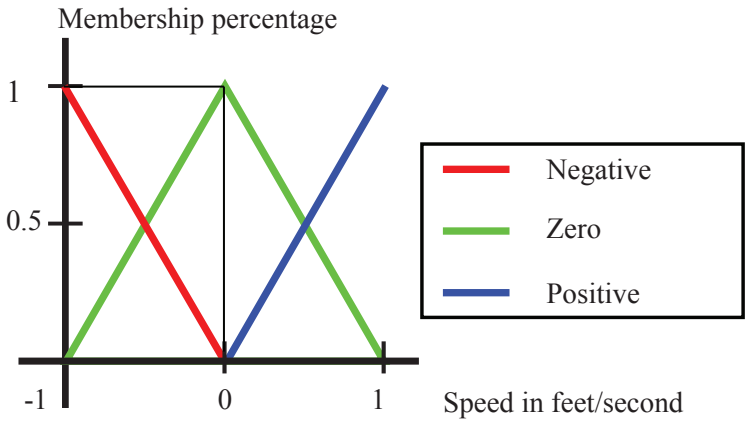

Figure 6.5: Speed input fuzzification example

By looking at the triangular membership functions the distance becomes 60% Medium Small and 40% Medium and 0% for all the other membership functions (inputs can take two membership functions because of the 50% overlap of the triangles). Symmetrical triangular membership functions of 50% overlap are used in this project. This setup is recommended in [1] and [8] to insure smooth output, simplification of the design and reduction of computational complexity. The whole range of inputs has been mapped onto fuzzy membership functions, this is important as we do not want an input with 0% membership to all the fuzzy sets. This could lead to a faulty output

command. Fuzzification of the speed input in figure 6.5 becomes 100% Zero, and 0% for Positive and Negative membership functions. Once fuzzification is done, the next step would be to find out what if-then statements apply to all the possible combination of the fuzzy sets obtained from the fuzzification stage.

Rule base is collection of the if-then statements that define the controller's action. They are set up by the control designer based on experience and the desired behavior of the system. In our design for the fuzzy position controller the if-then statements are summarized in figure 6.6. The rows present the membership functions of the speed, the columns are the membership functions of distance. The way to read the top left rule is: if distance is small and if speed is negative, then output is Small. The 'and' function between the two if statements translate to minimum function as will be explained in the rest of this example.

	Small	Medium Small	Medium	Medium Large	Large
Negative	Small	Medium	Medium	Large	Large
Zero	Small	Small	Medium	Medium	Large
Positive	Small	Small	Small	Medium	Large

Figure 6.6: Rule base for fuzzy position controller

The inference engine, the next processing step in this controller, finds the if-then rules that are fired (activated), and the firing strength. These terminologies will be explained further as the example continues. The way to obtain the firing strength is by taking the minimum value of the membership values in the specific fired rule. Remembering the membership values so far, for distance they are 60% Medium Small and 40% Medium and 0% for all the other membership functions. For speed the membership values are 100% Zero, and 0% for Positive and Negative membership functions. The inference engine finds the rules to be applied (all the possible combinations from the nonzero membership values). They are : if distance is Medium Small = 60% and Speed is Zero = 100% then output is Small, the other and final fired rule is if distance is Medium= 40% and Speed is Zero= 100% then output is Medium. The firing strength

for the first rule is calculated as follows: min (0.6 , 1) = 0.6. Then the firing strength for the output membership function Small is 0.6. The same is applied to the second rule, min (0.4, 1) = 0.4. Then the firing strength for the output membership function is 0.4. These two outputs are added together using a max function. This is shown in figure 6.7.

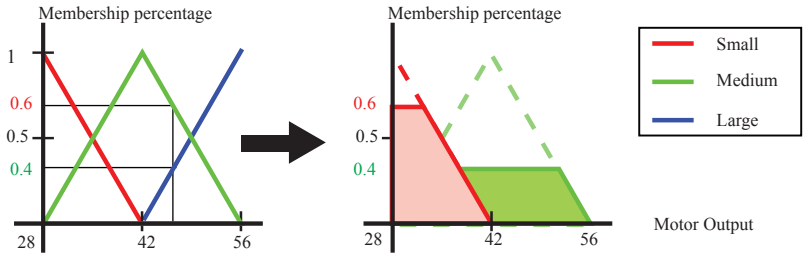

Figure 6.7: Fuzzy inference engine in action

The last step in the fuzzy logic controller output calculation is the defuzzification of the inference engine's output. This is done using the center of area method; this is shown in figure 6.8. The output of the defuzzification stage is passed onto the motor for action.

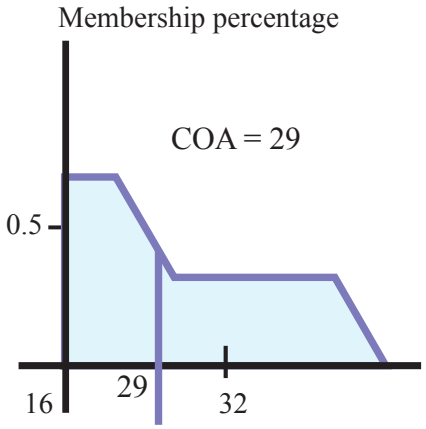

Figure 6.8: 3-parameter position controller example

6.2.2 Fuzzy orientation controller

This section shows the design parameters for the fuzzy orientation controller.

The membership functions for the inputs, angle relative to goal and angular speed is given in 6.9 and 6.10 respectively. The output membership function is shown in

6.11.

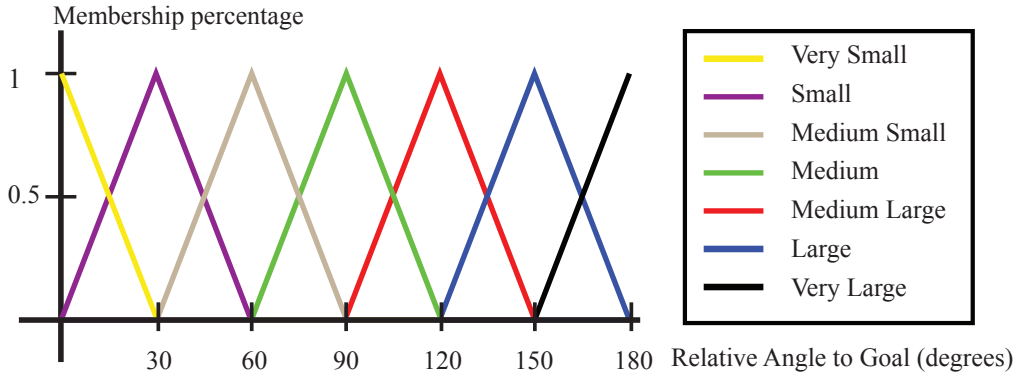

Figure 6.9: Fuzzy orientation angle input membership functions

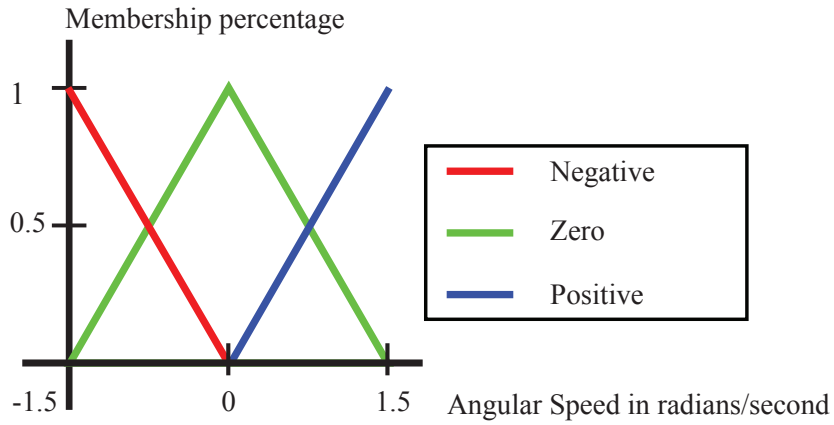

Figure 6.10: Fuzzy orientation angular speed input membership functions

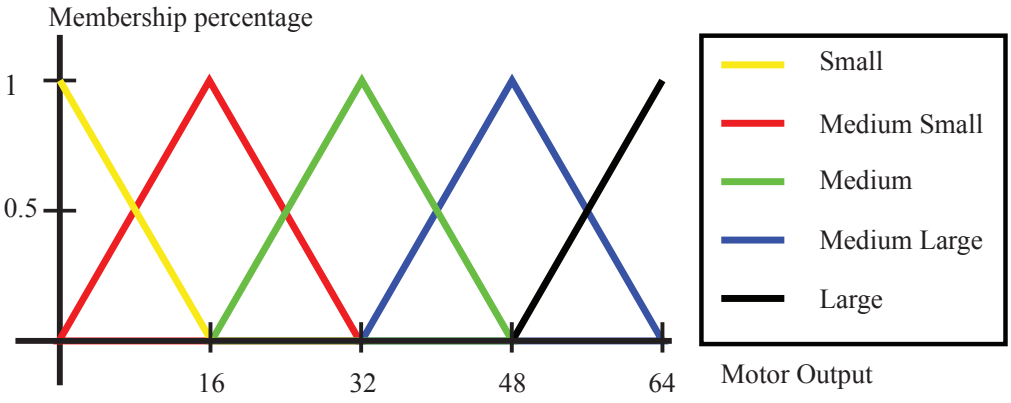

Figure 6.11: Fuzzy orientation controller output membership functions

The rule base is shown in figure 6.12.

	Very Small	Small	Medium Small	Medium	Medium Large	Large	Very Large
Negative	Medium Small	Medium	Medium	Medium Large	Medium Large	Large	Large
Zero	Medium Small	Medium Small	Medium	Medium	Medium Large	Large	Large
Positive	Small	Small	Small	Medium Small	Medium	Medium Large	Large

Figure 6.12: Rule base for fuzzy orientation controller

The defuzzification method used is again center of area.

6.3 Results and discussion

This section presents the results of twenty runs taken using the fuzzy logic controller discussed previously in this chapter. They are a mixture of runs with and without disturbances. Unfortunately we did not separate the runs into two classes: with disturbances and without disturbances.

Figure 6.13 shows the average of the x position vs. motor commands of the blimp run, the green line presents the average across the runs, and the black lines show one standard deviation from the mean. Figure 6.14 shows the average of the z position vs. motor commands of the blimp, the red line presents the average across the runs, and the pink lines show one standard deviation from the mean. Figure 6.15 shows the average of the y position (height) vs. motor commands of the blimp, the blue line presents the average across the runs, and the cyan lines show one standard deviation from the mean. Figure 6.16 finally shows the top view of the average lap taken by the blimp between the two points (39, 17) and (19, 17).

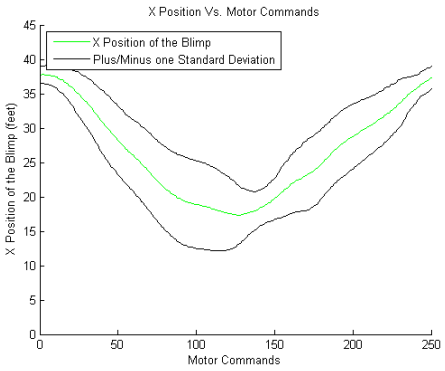

Figure 6.13: X position vs. Motor commands integrated with and without disturbances

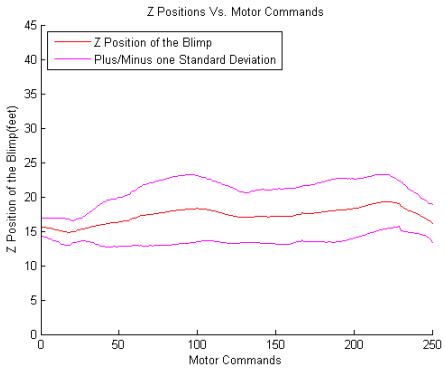

Figure 6.14: Z position vs. Motor commands integrated with and without disturbances

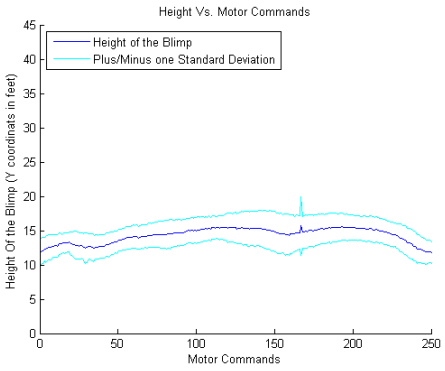

Figure 6.15: Height vs. Motor commands integrated with and without disturbances

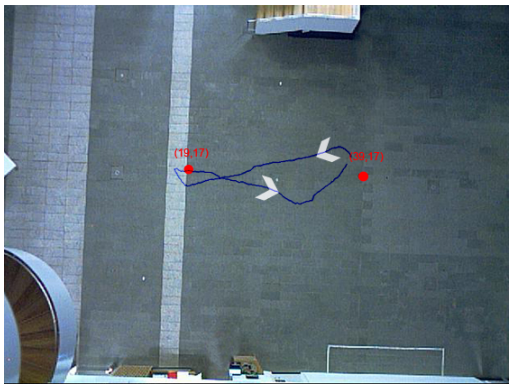

Figure 6.16: Z position vs. X position integrated with and without disturbances

Table 6.1 gives the performance measures calculated of the average run as well as the standard deviations from those averages. The area between the standard deviations is also calculated. Overall the controller has performed very well as can be seen in the very smooth plots of figures 6.13 through 6.15. The average run in figure 6.15 is smooth and heads towards goal directly, makes the 180-degree turn and goes to the second way point. The blimp moves in a curve towards the second way point, overshooting the 180-degree goal indicating that the orientation controller is more aggressive than it should be and may need more tuning to achieve a more direct path to the second way point. The next section compares the fuzzy logic controller with the previous controllers discussed in chapters 4 and 5.

Value	Average	Standard deviation
Time (minutes)	3.3512	0.9932
Average x (feet)	26.4366	6.6624
Average z (feet)	17.2974	1.1100
Average y (feet)	14.3055	1.0782
Area between x standard deviations	2135	
Area between z standard deviations	1862	
Area between y standard deviations	1003	
Area in 3D	1.9239e+003	987.5214

Table 6.1: Fuzzy logic controller results

Chapter 7

Conclusions and Future Work

This chapter provides a comparison between the three controllers applied to the robotic blimp. Table 7.1 provides a numerical summary of the results. The statistical significance work was done in this chapter using Student's t-test for unpaired samples, one-tailed hypothesis test.

Comparison metric	3 Parameter State based controller	4 parameter State based controller	Fuzzy logic controller
Area in 3D (feet^2)	2.7141e+003	2.3005e+003	1.9239e+003
Time (minute)	3.5538	4.2716	3.3512

Table 7.1: Performance metrics of the three controllers.

The average area in 3D for fuzzy logic controller is lower than that for 3-parameter state based controller (t =2.61377, DF=38.0, p<0.007); an improvement of 29.11%. The average area in 3D for fuzzy logic controller is also lower than that for 4-parameter state based controller (t =1.03713, DF=38.0, p<0.154); an improvement of 16.35%. This is not a very strong conclusion as the p<0.154 is not extremely significant. The average area in 3D for 4-parameter state based controller was lower than that for 3-parameter state based controller (t =1.23543 , DF=38.0, p<0.12)); an improvement of 15.26%. We can conclude that the fuzzy logic controller kept to the required path in its runs more than the other controllers.

In terms of time, the average time the fuzzy logic controller took to complete its run is lower than that for 3-parameter state based controller (t =1.07789, DF=38.0, p<0.15); an improvement of 5.7%. This is not a very strong conclusion as the p<0.15 is not extremely significant. The average time the fuzzy logic controller took to com-

plete its run is also lower than that for 4-parameter state based controller (t =2.51349, DF=38.0, p<0.01); an improvement of 21.55%. The average time for 4-parameter state based controller was 20.2% higher than that for 3-parameter State based controller (t =1.40210, DF=38.0, p<0.1). We can conclude that the fuzzy logic controller on average completed its task faster than both of the other controllers.

The set of figures below compares the average runs of the three controllers. The average for a given controller is calculated over all 20 runs, with and without disturbance conditions. Figure 7.1 below shows the perfect controller's run that is the standard for comparison. Figure 7.2 shows the average run (z position vs. x position) for the 3-parameter state based controller. Figure 7.3 shows the average run (z position vs. x position) for the 4-parameter state based controller. Figure 7.4 shows the average run (z position vs. x position) for the fuzzy logic controller. The 3-parameter state based controller was not able to head towards the first waypoint directly and went beyond that point on average as can be seen in figure 7.2. These problems were solved, however; the blimp failed to tightly follow the straight line that connected the two waypoints as can be seen in figure 7.3. The fuzzy logic followed the line connecting the two waypoints better as can be seen in figure 7.4.

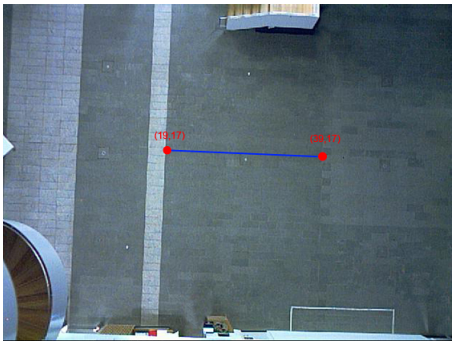

Figure 7.1: Perfect controller's average run for comparison

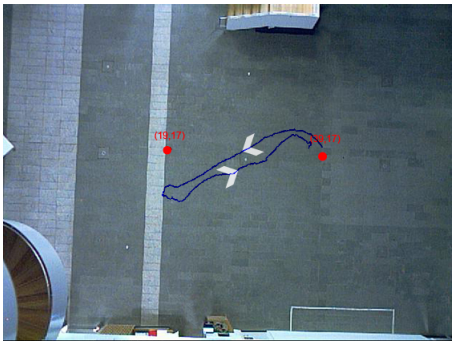

Figure 7.2: 3-parameter state based controller's average run

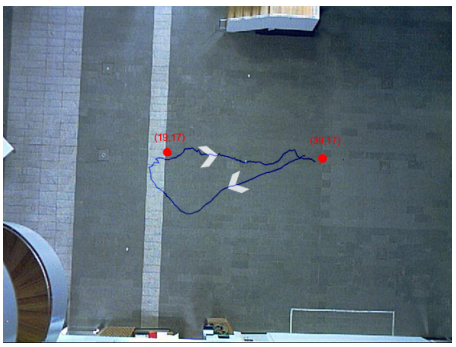

Figure 7.3: 4-parameter state based controller's average run

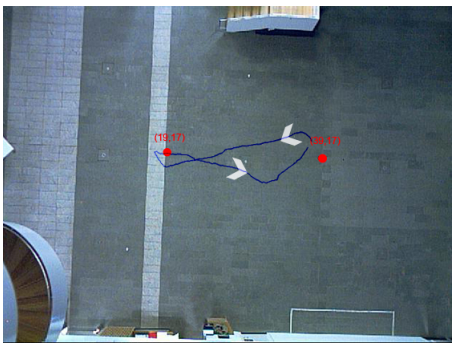

Figure 7.4: Fuzzy logic controller's average run

The continuous range of outputs the fuzzy logic controller provides, as well as the ability to fuzzify the inputs into more than one membership function have made a difference in the performance of the blimp. These factors have rendered the fuzzy logic controller to be the better performing controller.

Further tuning to the fuzzy controller would enhance the performance of the robotic blimp. This could be done manually by changing the number and/or shape of the membership functions. The ranges the membership functions cover can also be manipulated to tune the fuzzy logic controller. The rule base can always be changed to alter the behavior of the controller and is an important factor in tuning the controller. Finally, the defuzzification method can be changed, where the center of area method that is used throughout this project provides a weighted average output to the motors, a more/less aggressive method could be used. Some other defuzzification methods that can be tested are: Mean of maxima, leftmost maximum and rightmost maximum. Automatic tuning; however, can be done in the form of ANFIS (adaptive neural fuzzy inference system)[17]. Further work can also be done through obtaining a mathematical model of the robotic blimp and applying computer simulations to understand the best way to apply control algorithm on this platform.

Appendix A

3-Parameter State Based Controller Action List: Orientation

This appendix lists the 27 actions that can be taken by the robotic blimp when operating under the 3-parameter state based orientation controller. The actions are given in table A.1. The following paragraphs present the abbreviations used in table A.1, to understand the ranges of the linguistic variables please refer to chapter 4.

The present state and future state can take any of the following values: clockwise (cw), anticlockwise (acw) and near. The speed state can take any of the following values: fast clockwise (fcw), fast anticlockwise (facw) and slow.

The action the controller takes can be any of the following: accelerate clockwise (C), accelerate clockwise fast (CC), do nothing (N), accelerate anticlockwise (A), accelerate anticlockwise fast (AA).

There are four special functions in the orientation controller. These special functions output a controller command only after looking at the current angular velocity of the blimp. The first special function (special function 1) is as follows: if fcw > 0.08 rad/second then action is AA else use A. The second special function (special function 2) is as follows: if facw > 0.08 rad/second then action is C else use N. The third special function (special function 3): if facw > 0.08 rad/second then action is CC else use C. The fourth special function (special function 4): if fcw > 0.08 rad/second then action is A else use N. The '*' used in table A.1 below indicates a state that should not occur, the action in this case is N.

Present orientation relative to goal	Future orientation relative to goal	Speed direction relative to goal	Action
cw	cw	fcw	Special function 1
cw	cw	facw	Special function 2
cw	cw	slow	A
cw	acw	fcw	N
cw	acw	facw	Special function 3
cw	acw	slow	A
cw	near	fcw	*
cw	near	facw	Special function 3
cw	near	slow	N
acw	cw	fcw	Special function 1
acw	cw	facw	N
acw	cw	slow	C
acw	acw	fcw	Special function 4
acw	acw	facw	Special function 3
acw	acw	slow	C
acw	near	fcw	Special function 1
acw	near	facw	*
acw	near	slow	N
near	cw	fcw	Special function 1
near	cw	facw	*
near	cw	slow	A
near	acw	fcw	*
near	acw	facw	Special function 3
near	acw	slow	C
near	near	fcw	Special function 1
near	near	facw	Special function 3
near	near	slow	N

Table A.1: 3-parameter state orientation controller action list

Appendix B

3-Parameter State Based Controller Action List: Position

This appendix lists the 27 actions that can be taken by the robotic blimp when operating under the 3-parameter state based position controller. The actions are given in table B.1. The following paragraphs present the abbreviations used in the table, to understand the ranges of the linguistic variables please refer to chapter 4.

The present state and future state can take any of the following values: front (fnt), beyond (byd) and near. The speed state can take any of the following values: fast forward (ffwd), fast backward (fbck) and slow.

The actions the controller takes can be any of the following: accelerate forward (F), accelerate forward fast (FF), do nothing (N), accelerate reverse (R), accelerate reverse fast (RR).

There are two special functions in the position controller. These special functions output a controller command only after looking at the current linear velocity of the blimp. The first special function (special function f) is as follows: if fbck > 0.08 rad/second then action is FF else use F. The second special function (special function r) is as follows: if ffwd > 0.08 rad/second then action is RR else use R. The '*' used in table B.1 below indicates a state that should not occur, the action in this case is N.

Present orientation relative to goal	Future orientation relative to goal	Speed direction relative to goal	Action
fnt	fnt	ffwd	N
fnt	fnt	fbck	special function f
fnt	fnt	slow	F
fnt	byd	ffwd	special function r
fnt	byd	fbck	*
fnt	byd	slow	R
fnt	near	ffwd	special function r
fnt	near	fbck	*
fnt	near	slow	N
byd	fnt	ffwd	*
byd	fnt	fbck	special function f
byd	fnt	slow	F
byd	byd	ffwd	special function r
byd	byd	fbck	N
byd	byd	slow	R
byd	near	ffwd	*
byd	near	fbck	special function f
byd	near	slow	N
near	fnt	ffwd	*
near	fnt	fbck	special function f
near	fnt	slow	F
near	byd	ffwd	special function r
near	byd	fbck	*
near	byd	slow	R
near	near	ffwd	special function r
near	near	fbck	special function f
near	near	slow	N

Table B.1: 3-parameter state position controller action list

Appendix C

4-Parameter State Based Controller Action List: Orientation

This appendix lists the 81 actions that can be taken by the robotic blimp when operating under the 4-parameter state based orientation controller. The actions are given in table C.1 are the actions taken if the previous state is clockwise (cw). Table C.2 lists the actions if the previous state was anticlockwise (acw). Finally table C.3 lists the actions if the previous state was near. The following paragraphs presents the abbreviations used in the table, to understand the ranges of the linguistic variables please refer to chapter 5.

The present state, future state and previous state can take any of the following values: clockwise (cw), anticlockwise (acw) and near. The speed state can take any of the following values: fast clockwise (fcw), fast anticlockwise (facw) and slow.

The action the controller takes can be any of the following: accelerate clockwise (C), accelerate clockwise fast (CC), do nothing (N), accelerate anticlockwise (A), accelerate anticlockwise fast (AA).

There are four special functions in the orientation controller. These special functions output a controller command only after looking at the current angular velocity of the blimp. The first special function (special function 1) is as follows: if fcw > 0.08 rad/second then action is AA else use A. The second special function (special function 2) is as follows: if facw > 0.08 rad/second then action is C else use N. The third special function (special function 3): if facw > 0.08 rad/second then action is CC else use C. The fourth special function (special function 4): if fcw > 0.08 rad/second then action is A else use N. The '*' used in tables C.1,C.2 and C.3 below indicates a state that should not occur, the action in this case is N.

Previous orientation relative to goal	Present orientation relative to goal	Future orientation relative to goal	Speed direction relative to goal	Action
cw	cw	cw	fcw	Special function 1
cw	cw	cw	facw	Special function 2
cw	cw	cw	slow	A
cw	cw	acw	fcw	N
cw	cw	acw	facw	Special function 3
cw	cw	acw	slow	A
cw	cw	near	fcw	*
cw	cw	near	facw	Special function 3
cw	cw	near	slow	N
cw	acw	cw	fcw	Special function 1
cw	acw	cw	facw	N
cw	acw	cw	slow	C
cw	acw	acw	fcw	C
cw	acw	acw	facw	Special function 3
cw	acw	acw	slow	C
cw	acw	near	fcw	Special function 1
cw	acw	near	facw	*
cw	acw	near	slow	N
cw	near	cw	fcw	Special function 1
cw	near	cw	facw	*
cw	near	cw	slow	A
cw	near	acw	fcw	*
cw	near	acw	facw	Special function 3
cw	near	acw	slow	C
cw	near	near	fcw	Special function 1
cw	near	near	facw	Special function 3
cw	near	near	slow	N

Table C.1: 4-parameter state controller action list when previous state = clockwise

Previous orientation relative to goal	Present orientation relative to goal	Future orientation relative to goal	Speed direction relative to goal	Action
acw	cw	cw	fcw	Special function 1
acw	cw	cw	facw	Special function 3
acw	cw	cw	slow	Special function 1
acw	cw	acw	fcw	*
acw	cw	acw	facw	Special function 3
acw	cw	acw	slow	C
acw	cw	near	fcw	*
acw	cw	near	facw	Special function 3
acw	cw	near	slow	N
acw	acw	cw	fcw	Special function 1
acw	acw	cw	facw	*
acw	acw	cw	slow	C
acw	acw	acw	fcw	Special function 4
acw	acw	acw	facw	Special function 2
acw	acw	acw	slow	C
acw	acw	near	fcw	Special function 4
acw	acw	near	facw	*
acw	acw	near	slow	N
acw	near	cw	fcw	Special function 1
acw	near	cw	facw	*
acw	near	cw	slow	A
acw	near	acw	fcw	*
acw	near	acw	facw	Special function 3
acw	near	acw	slow	C
acw	near	near	fcw	Special function 4
acw	near	near	facw	Special function 2
acw	near	near	slow	N

Table C.2: 4-parameter state controller action list when previous state = anticlockwise

Previous orientation relative to goal	Present orientation relative to goal	Future orientation relative to goal	Speed direction relative to goal	Action
near	cw	cw	fcw	Special function 2
near	cw	cw	facw	N
near	cw	cw	slow	Special function 4
near	cw	acw	fcw	N
near	cw	acw	facw	Special function 3
near	cw	acw	slow	N
near	cw	near	fcw	*
near	cw	near	facw	Special function 3
near	cw	near	slow	N
near	acw	cw	fcw	Special function 1
near	acw	cw	facw	N
near	acw	cw	slow	C
near	acw	acw	fcw	Special function 1
near	acw	acw	facw	Special function 2
near	acw	acw	slow	C
near	acw	near	fcw	Special function 4
near	acw	near	facw	*
near	acw	near	slow	N
near	near	cw	fcw	Special function 1
near	near	cw	facw	*
near	near	cw	slow	A
near	near	acw	fcw	*
near	near	acw	facw	Special function 3
near	near	acw	slow	C
near	near	near	fcw	Special function 4
near	near	near	facw	Special function 3
near	near	near	slow	N

Table C.3: 4-parameter state controller action list when previous state = near

Appendix D

4-Parameter State Based Controller Action List: Position

This appendix lists the 81 actions that can be taken by the robotic blimp when operating under the 4-parameter state based position controller. The actions are given in table D.1, D.2 and D.3. The following paragraphs presents the abbreviations used in the table, to understand the ranges of the linguistic variables please refer to chapter 5.

The previous state, present state and future state can take any of the following values: front (fnt), beyond (byd) and near. The speed state can take any of the following values: fast forward (ffwd), fast backward (fbck) and slow.

The actions the controller takes can be any of the following: accelerate forward (F), accelerate forward fast (FF), do nothing (N), accelerate reverse (R), accelerate reverse fast (RR).

There are two special functions in the orientation controller. These special functions output a controller command only after looking at the current linear velocity of the blimp. The first special function (special function f) is as follows: if fbck > 0.08 rad/second then action is FF else use F. The second special function (special function r) is as follows: if ffwd > 0.08 rad/second then action is RR else use R. The '*' used in table D.1, D.2 and D.3 below indicates a state that should not occur, the action in this case is N.

Previous orientation relative to goal	Present orientation relative to goal	Future orientation relative to goal	Speed direction relative to goal	Action
fnt	fnt	fnt	ffwd	N
fnt	fnt	fnt	fbck	special function f
fnt	fnt	fnt	slow	F
fnt	fnt	byd	ffwd	special function r
fnt	fnt	byd	fbck	*
fnt	fnt	byd	slow	R
fnt	fnt	near	ffwd	special function r
fnt	fnt	near	fbck	*
fnt	fnt	near	slow	N
fnt	byd	fnt	ffwd	*
fnt	byd	fnt	fbck	special function f
fnt	byd	fnt	slow	*
fnt	byd	byd	ffwd	special function r
fnt	byd	byd	fbck	N
fnt	byd	byd	slow	R
fnt	byd	near	ffwd	*
fnt	byd	near	fbck	special function f
fnt	byd	near	slow	N
fnt	near	fnt	ffwd	*
fnt	near	fnt	fbck	special function f
fnt	near	fnt	slow	F
fnt	near	byd	ffwd	special function r
fnt	near	byd	fbck	*
fnt	near	byd	slow	R
fnt	near	near	ffwd	N
fnt	near	near	fbck	special function f
fnt	near	near	slow	F

Table D.1: 4-parameter state controller action list when previous state = fnt

Previous orientation relative to goal	Present orientation relative to goal	Future orientation relative to goal	Speed direction relative to goal	Action
byd	fnt	fnt	ffwd	F
byd	fnt	fnt	fbck	special function fr
byd	fnt	fnt	slow	F
byd	fnt	byd	ffwd	special function r
byd	fnt	byd	fbck	*
byd	fnt	byd	slow	*
byd	fnt	near	ffwd	special function r
byd	fnt	near	fbck	*
byd	fnt	near	slow	F
byd	byd	fnt	ffwd	*
byd	byd	fnt	fbck	special function f
byd	byd	fnt	slow	F
byd	byd	byd	ffwd	special function r
byd	byd	byd	fbck	N
byd	byd	byd	slow	R
byd	byd	near	ffwd	*
byd	byd	near	fbck	special function f
byd	byd	near	slow	N
byd	near	fnt	ffwd	*
byd	near	fnt	fbck	special function f
byd	near	fnt	slow	F
byd	near	byd	ffwd	special function r
byd	near	byd	fbck	*
byd	near	byd	slow	R
byd	near	near	ffwd	N
byd	near	near	fbck	special function f
byd	near	near	slow	R

Table D.2: 4-parameter state controller action list when previous state = byd

Previous orientation relative to goal	Present orientation relative to goal	Future orientation relative to goal	Speed direction relative to goal	Action
near	fnt	fnt	ffwd	F
near	fnt	fnt	fbck	special function f
near	fnt	fnt	slow	special function f
near	fnt	byd	ffwd	special function r
near	fnt	byd	fbck	*
near	fnt	byd	slow	*
near	fnt	near	ffwd	special function r
near	fnt	near	fbck	*
near	fnt	near	slow	F
near	byd	fnt	ffwd	*
near	byd	fnt	fbck	special function f
near	byd	fnt	slow	*
near	byd	byd	ffwd	special function r
near	byd	byd	fbck	N
near	byd	byd	slow	R
near	byd	near	ffwd	*
near	byd	near	fbck	special function f
near	byd	near	slow	N
near	near	fnt	ffwd	*
near	near	fnt	fbck	special function f
near	near	fnt	slow	F
near	near	byd	ffwd	special function r
near	near	byd	fbck	*
near	near	byd	slow	R
near	near	near	ffwd	special function r
near	near	near	fbck	special function f
near	near	near	slow	N

Table D.3: 4-parameter state controller action list when previous state = near

Bibliography

[1] Gerrit Christiaan Avenant. Autonomous flight control system for an airship. Master's thesis, Stellenbosch University, 2010.

[2] J. R. Azinheira, P. Rives, J. R. H. Carvalho, G. F. Silveira, E. C. de Paiva, and S. S. Bueno. Visual servo control for the hovering of all outdoor robotic airship. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA '02*, volume 3, pages 2787–2792, 2002.

[3] J.R. Azinheira, E.C. de Paivab, J.J.G. Ramosb, and S.S. Buenob. Hovering control of an autonomous unmanned airship. In *4th Portuguese Conference on Automatic Control*, pages 430–435, 2000.

[4] A. Elfes, J.F. Montgomery, J.L. Hall, S.S. Joshi, J. Payne, and C.F. Bergh. Autonomous flight control for a titan exploration aerobot. In *Robotics Science and Systems*, 2005.

[5] M. Falahpour, H. Moradi, H. H. Refai, M. Atiquzzaman, R. Rubin, and P. G. LoPresti. Performance comparison of classic and fuzzy logic controllers for communication airships. In *Proc. IEEE/AIAA 28th Digital Avionics Systems Conf. DASC '09*, pages 4.A.6–1 – 4.A.6–8, 2009.

[6] S. B. V. Gomes and Jr. Ramos, J. G. Airship dynamic modeling for autonomous operation. In *Proc. IEEE Int Robotics and Automation Conf*, volume 4, pages 3462–3467, 1998.

[7] P. Gonzalez, W. Burgard, R. Sanz, and Fernandez. J.L. Developing a low-cost autonomous indoor blimp. In *Journal of Physical Agents*, volume 3, pages 43–52, January, 2009.

[8] G. Hill, E. Horstkotte, and J. Teichrow. *Fuzzy-C Development System User's Manual*. Togai InfraLogic Inc., Irvine, CA, release 2.1 edition, June 1989.

[9] E. Hygounenc, I. Jung, P. Soueres, and S. Lacroix. The autonomous blimp project of laas-cnrs: Achievements in flight control and terrain mapping. In *The International Journal of Robotics Research*, volume 23, pages 473–511, 2004.

[10] E. Hygounenc and P. Soueres. Automatic airship control involving backstepping techniques. In *Proc. IEEE Int Systems, Man and Cybernetics Conf*, volume 6, page 6, 2002.

[11] E. Hygounenc and P. Soueres. Lateral path following gps-based control of a small-size unmanned blimp. In *Proc. IEEE Int. Conf. Robotics and Automation ICRA '03*, volume 1, pages 540–545, 2003.

[12] Emmanuel Hygounenc, Il-Kyun Jung, Philippe Soueres, and Simon Lacroix. The autonomous blimp project of laas-cnrs: Achievements in flight control and terrain mapping. *The International Journal of Robotics Research*, 23:473–511, 2004.

[13] Jan Jantzen. Design of fuzzy controllers. Technical report, Technical Universtiy of Denmark, 1998.

[14] Guo Jian-guo and Zhou Jun. Altitude control system of autonomous airship based on fuzzy logic. In *Proc. 2nd Int. Symp. Systems and Control in Aerospace and Astronautics ISSCAA 2008*, pages 1–5, 2008.

[15] T. Kampke and A. Elfes. Optimal aerobot trajectory planning for wind-based opportunistic flight control. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2003)*, volume 1, pages 67–74, 2003.

[16] G. A. Khoury and Gillett. *Airship Technology*. Cambridge University, Cambridge, England, 2002.

[17] Wong Wei Kitt, Ali Chekima, and Jamal A. Dhargam. Design and verification of an anfis based control system for an autonomous airship on a simulation environment. *GJCAT*, 1:119–.128, 2011.

[18] A. Kornienko. *System Identification Approach for Determining Flight Dynamical Characteristics of an Airship from Flight Data*. PhD thesis, University of Stuttgart, 2006.

[19] E.A. Kulczycki, S.S. Joshi, and R.A. Hess. Towards controller design for autonomous airships using slc and lqr methods. In *AIAA Guidance Navigation and Control Conference and Exhibit*, 2006.

[20] P. Kungl, M. Schlenker, D.A. Wimmer, and B.H. Kröplin. Instrumentation of remote controlled airship "lotte" for in-flight measurements. In *Aerospace Science and Technology*, volume 8, pages 599–610, 2004.

[21] S. Lacroix and Il-Kyun Jung. High resolution terrain mapping with an autonomous blimp. In *Proc. IEEE/RSJ Int Intelligent Robots and Systems Conf*, volume 1, pages 781–786, 2002.

[22] Simon Lacroix, Il-Kyun Jung, Philippe Soueres, Emmanuel Hygounenc, and Jean-Paul Berry. The autonomous blimp project of laas/cnrs :current status. In Bruno Siciliano and Paolo Dario, editors, *Experimental Robotics VIII*, volume 5 of *Springer Tracts in Advanced Robotics*, pages 487–496. Springer Berlin / Heidelberg, 2003.

[23] Yiwei Liu, Zengxi Pan, D. Stirling, and F. Naghdy. Control of autonomous airship. In *Proc. IEEE Int Robotics and Biomimetics (ROBIO) Conf*, pages 2457–2462, 2009.

[24] T. Lutz, P. Funk, A. Jakobi, and S. Wagner. Summary of aerodynamic studies on the lotte airship. In *4th International Airship Convention and Exhibition*, pages 1–12, 2002.

[25] A. Ntelidakis. Using a blimp to build an interior model of the forum. Master's thesis, School of Informatics, University of Edinburgh, 2010.

[26] A. Ollero and L. Merino. Control and perception techniques for aerial robotics. In *Annual reviews in control 28*, pages 167–178, 2004.

[27] J. Payne and S.S. Joshi. 6 degree-of-freedom non-linear robotic airship model for autonomous control. Technical report, University of California, 2004.

[28] J.J.G. Ramos, E.C. de Paiva, J.R. Azinheira, S.S. Bueno, Bergermana M., P.A.V. Ferreira, and J.R.H. Carvalho. Lateral/directional control for an autonomous unmanned airship. In *Aircraft Engineering and Aerospace Technology*, volume 73, pages 453–458, 2001.

[29] Toshihiko Takaya, Hidenori Kawamura, Yoshihiro Minagawa, Masahito Yamamoto, and Azuma Ohuchi. Pid landing orbit motion controller for an indoor blimp robot. *Artificial Life and Robotics*, 10:177–184, 2006.

[30] C. Weisbin, J. Mrozinski, H. Hua, K. Shelton, J. H. Smith, A. Elfes, W. Lincoln, V. Adumitroaie, and R. Silberg. Human-robot lunar exploration:pressurized vs. unpressurized rovers. In *Proc. 19th Int. Conf. Systems Engineering ICSENG '08*, pages 8–12, 2008.

[31] D. Wimmer, M. Bildstein, K.H. Well, M. Schlenker, P. Kungl, and B.-H. Kröplin. Research airship "lotte" development and operation of controllers for autonomous flight phases. In *International Conference on Intelligent Robots and Systems*, 2002.

[32] Online: www.mineseeker.com; Date accessed: 17th August 2011.