VISION BASED CONTROL OF AN AUTONOMOUS BLIMP WITH ACTUATOR SATURATION USING PULSE WIDTH MODULATION

A thesis Presented

by

Korkut Bekiroglu

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Northeastern University
Boston, Massachusetts

June 2010
“If anyone travels on a road in search of knowledge, Allah will cause him to travel on one of the roads of Paradise. The angels will lower their wings in their great pleasure with one who seeks knowledge, the inhabitants of the heavens and the Earth and the fish in the deep waters will ask forgiveness for the learned man. The superiority of the learned man over the devout is like that of the moon, on the night when it is full, over the rest of the stars. The learned are the heirs of the Prophets, and the Prophets leave neither dinar nor dirham, leaving only knowledge, and he who takes it takes an abundant portion.”

The Prophet Muhammad
The purpose of this study was tended toward applied research. This thesis demonstrated an effective way to track any defined object with unmanned aerial vehicle (UAV) or lighter than air vehicle by using Continuously Adaptive Mean Shift (Camshift) Algorithm, Proportional-Derivative (PD) controller and the Pulse Width modulation (PWM).

The thesis describes a technique in which the blimp, with a wireless video camera capturing the image to a computer, and radio controlled propellers are used to move the blimp to track a target object. The code for the algorithm, written in Matlab, captures the frames and tracks the object to find the co-ordinates of the center of the target object and sends the required pulses to the blimp through the remote controller of the blimp.

The verification of the mathematical model of the blimp is also presented in order to design a controller for the model. During the identification process, it was realized that the system has saturation nonlinearities on its velocity and actuators. After validating the model, controller design procedure was also presented for the validated model of the system. Additionally, Pulse Width Modulation techniques were used to adjust the speed of the blimp and overcome the saturation nonlinearities of the system after the implementation of the PD controller.
Acknowledgements

First and foremost, I would like to extend my infinite thanks to my Dear LORD, who is the most gracious, and the most merciful. Few words can express the gratitude, I feel from the depth of my heart, for His countless bounties.

There are also many people who have provided me with valuable guidance, technically as well as emotionally which has contributed in the final making of this thesis. I would like continue by sincerely thanking Prof. Mario Sznaier and Prof. Bahram Shafai for giving me the opportunity to work on this thesis and providing the facilities. They have been very willing to provide help whenever I needed. They have provided me with great role models and great sources of inspiration to achieve academic success. In addition, I would like to thank the Robust System Laboratory and Prof. Mario Sznaier for providing the funding necessary to complete this project.

I would like to thank Mustafa Ayazoglu and Necmiye Ozay all of the help and explanations. They gave me a lot of guidance from the theory to the implementation of the blimp control. Furthermore, they have always been there to help me with their support and encouragement.

I would like to express the deepest appreciation to my family for all of the help, support, and guidance that they have given me throughout my academic career. I would also thank to my friends whom have given me such wonderful support and help. I thank you and love you all.

Finally, I would like to thank my wife, Saliha, for all of her love and support and for putting up with me through all of the stressful times. I thank you very much Saliha and love you with all my heart.
# Contents

Abstract ii

Acknowledgements iii

List of Figures vi

List of Tables vii

1 Introduction 1
   1.1 Background ...................................................... 1
   1.2 Related Works .................................................. 1
   1.3 Thesis Overview ............................................... 3

2 Identification and Controller Design 6
   2.1 System Overview ................................................ 6
   2.1.1 The Mechanical Design of The Blimp ....................... 6
   2.1.2 How the System Works ........................................ 8
   2.2 Identification .................................................. 9
   2.3 Controller Design ............................................... 14
   2.3.1 General Structure of Nonlinear Systems .................... 14
   2.3.2 Controller Design Techniques and Stability Analysis for Non-
       linear Systems ................................................... 15
   2.3.3 PD Controller Design for The Blimp ....................... 17

3 System Structure, Hardware and Software 26
   3.1 System Hardware ................................................ 26
   3.1.1 Transmitter-Receiver Circuits and Solid State Relay ...... 26
   3.1.2 Parallel Port ................................................. 28
   3.1.3 Blimp-Gondola and Propellers ................................ 29
   3.1.4 The Wireless Camera and Computer ......................... 30
   3.2 System Software ................................................ 31
   3.2.1 How to Set Up The Program .................................. 31
   3.2.2 How to Use GUI .............................................. 33
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>System Overview</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Block Diagram of Visual Tracking System</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Effect of Aerodynamic Force</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Model of The Blimp</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>Interpolated Plot</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Interpolated Plot</td>
<td>14</td>
</tr>
<tr>
<td>2.7</td>
<td>Feedback Nonlinearity</td>
<td>15</td>
</tr>
<tr>
<td>2.8</td>
<td>Controller Implementation</td>
<td>18</td>
</tr>
<tr>
<td>2.9</td>
<td>Root Locus Uncompensated System</td>
<td>23</td>
</tr>
<tr>
<td>2.10</td>
<td>Root Locus compensated System</td>
<td>24</td>
</tr>
<tr>
<td>2.11</td>
<td>Step Response Compensated System</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Transmitter circuit</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Relay Circuit</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Port-Transmitter</td>
<td>29</td>
</tr>
<tr>
<td>3.4</td>
<td>Camera-Receiver</td>
<td>31</td>
</tr>
<tr>
<td>3.5</td>
<td>Graphical User Interface</td>
<td>32</td>
</tr>
<tr>
<td>4.1</td>
<td>Block Diagram of Color Object Tracking</td>
<td>40</td>
</tr>
<tr>
<td>4.2</td>
<td>Percentage level of PWMX</td>
<td>44</td>
</tr>
<tr>
<td>4.3</td>
<td>Percentage level of PWMY</td>
<td>45</td>
</tr>
<tr>
<td>4.4</td>
<td>Block Diagram of General Algorithm</td>
<td>46</td>
</tr>
</tbody>
</table>
# List of Tables

3.1 Parallel Port Output Pins Control. ........................................ 29
4.1 Reference Input. .............................................................. 41
4.2 Percentage level of PWM ................................................... 42
Dedicated to My Family...
Chapter 1

Introduction

1.1 Background

There are various application areas for unmanned aerial vehicles (UAV), not only for military applications, but also for civilian use. UAVs can be used for number of specific areas in military. It has low speed and low altitude allowing it to be advantageous for surveillance and monitoring. It can be used for ground traffic control, for observing and predicting rush hour traffic, for taking precautions measures to ease incidental situations. Moreover, it also has potential applications in fighting forest fires. A blimp fitted with either a camera or thermal sensors or a combination of both is able to survey affected areas and facilitate in the fight against them. Other possible applications include atmospheric studies and study of irrigated crop lands to investigate water supply distribution. In order to provide higher level of safety and detect a particular object, altitude of a blimp should be adjusted automatically using a visual sensor [1, 2]. Generally, the blimp tries to automatically detect the target object and avoid the obstacles to complete its given task. This is exactly the analogous to what pilots do in air.

1.2 Related Works

UAV or remotely piloted vehicle has been studied for several different areas including some experimental research. It is tremendously difficult to manage because of its great inertia and physical structure. Due to its challenging system dynamics only
one controller action would not alone suffice to control it. In order to overcome these issues, the pulse width modulation block was added to drive the motors. Gonzalez and his coworkers describe the design of an autonomous blimp-based controller [3]. It is crucial that we choose the optimal hardware to obtain maximum performance along with cost efficiency. They developed a simple and safe model for self-controlled blimp. While establishing the hardware, the advantages of additional physical information were taken from the data from the Gonzalez paper. Since the structure of the hardware directly affects the system dynamics, this is also important for robustness of the system.

Additionally, there are various different applications of hardware selection in the literature. Takanori and his colleagues developed an autonomous blimp for a surveillance systems in which they tried to design a blimp that can hover around a specific given target with a camera. They took advantage of Lucas and Kanada algorithm for detection and tracking the target object. They also developed different tracking algorithms for an aerial blimp robot. They also utilized motion segmentation to improve automatic detection and tracking of target object [4, 5]. Another visual tracking application of blimp was made by Seungyong. They fixed a camera on the floor in order to easily follow the blimp with a real time visual tracking algorithm with an image projection. They obtained positional information of the blimp from the camera [6]. In this work, the camera was mounted on the gondola, which enlarged the moving space of the blimp.

Recent research in visual feedback control of the unmanned blimp was done by Yasunori and his colleagues. They also used a fixed camera on the floor in order to obtain the coordinates of the blimp. This camera was used as a sensor on the feedback loop. The system dynamics were represented as a linear parameter varying system. This was then used for the design of the control Lyapunov function for this particular linear parameter varying system. In this setting, fixed camera still restricts the motility of the blimp and thus affects the stability of system for our case as well. Field of view of camera becomes effective on the control algorithm in this blimp research [7].

In addition to controller and tracker design, navigation and identification of UAV is also gaining importance in order to improve the tracker performance. Computer vision based Navigation for autonomous blimps are becoming more and more important in aforementioned application fields. Celho and his coworkers developed a navigation system extensively using computer vision algorithms. The first and most
important step for a computer vision based control system is gathering the data for estimating the position of the blimp relative to the target object. They developed an algorithm for the blimp to follow precise trajectories based on environmental visual information acquired by an on-board camera [8]. In addition, every single application will need a different capability to be able to complete its mission, therefore algorithms can vary for every application. There is also a research done to identify the blimp using input/output relations with the Carathodory-Fejer interpolation algorithm [9].

A tracking algorithm for the UAV was developed by using camshift as an indoor application in addition to these mentioned. A Pulse Width Modulation (PWM) algorithm was implemented between controller and the blimp in order to improve the stability and the performance of the system. This is mainly due to the nonlinearity off its velocity state. Furthermore, propellers work in two-states ON or OFF, therefore the PWM has to develop to make the velocity controls. We also tried to find an approximate model for this blimp system. Input/output relationships were used to verify the mathematical model of the system formulated using Newtonian laws of motion. The camera on the feedback path was mounted on the gondola with a specific angle. This is to collect image data with the purpose of acquiring the positional information of the blimp relative to the target object. Moreover, PWM algorithm is a rather smart approach to overcome the nonlinearity, large inertia of the blimp and speed control for the ON/OFF motors. Finally, the PWM increases the robustness of the overall system.

1.3 Thesis Overview

The goal of this work is to design and implement a controller for an unmanned blimp or lighter than air vehicle to track a selected object inside of a room, using only horizontal motion. Note that we assume that the blimp is already balanced and hovering in the air so there is no need to deal with the vertical motion. In figure 2.1 the overall system structure can be seen.

The organization of this thesis is as follows;

Chapter 2 details the identification of the blimp and controller design of the identified model. At the beginning, a briefly explanation will be represented to show choosing of the necessary hardware for control of the blimp. Then, the identification
process will be detailed that consists of two components. First of all, the gathering, interpolation and plotting of the gathered image data will be explained. Secondly, the mathematical model of the blimp will be described in order to compare it with the interpolated data to decide the final approximate model of the blimp. Afterwards, the effect of the aerial dynamic force on the system dynamics will briefly explained and a appropriate model of the blimp will also be decided. The importance of choosing of the hardware will also be investigated on the blimp dynamics. Consequently, the system dynamics and nonlinearity will be modeled which come along with the air drag. After briefly explaining the controller design methods for this kind of nonlinear system, a digital controller will be implemented with pole placement method in order to track the target object by using the discrete time root locus of the system. We will also specify why we used PWM and how it overcame the nonlinearity in the model.

Chapter 3 describes the software and the hardware used to collect the experimental data needed to identify and drive the system. Firstly, the system structure will be clarified. Secondly, all properties of every single hardware and the connection of the transmitter and the parallel port will be explained in detail. The hardware used to transmit commands and receive data will be discussed as well as the wireless camera. Furthermore, we will put more insight into the task of each hardware in the system. The properties of the camera to obtain necessary sampling rate will be also specified. The set up process of the software will be mentioned in detail as well. Then, we will explain how the GUI can be used to manually or automatically control the blimp as well as tracking software and Camshift and explain the functioning buttons on the GUI.

Chapter 4 will explain the camshift algorithm, controller and PWM. Initially, The general structure of the tracking algorithms, how they work and their application fields will be detailed in detail of the chapter 4. One of these tracking algorithms is the camshift -used for this thesis as well- will also be detailed. Prior to explaining camshift algorithm, a background on the mean-shift algorithm will be given. In addition to mean-shift algorithm, the camshift algorithm will be detailed and how it was applied to the blimp system. Next part of the chapter will cover all details for the controller and the PWM algorithm. The controller implementation will be outlined to the system in order to close the loop. Then, the general structure of the PWM and its implementation to the system will also be explained in detail. We will also reason how we decided on all of the parameters of the PWM and how it runs in the code.
Chapter 5 will discuss how far the automatic control of UAV was developed as well as what is still needed as a modification in order to improve its robustness and velocity. There will be brief explanation, in detail, all of the failures and successes that we were able to carry out in blimp control system. Additionally, some insights for the future implications will also tried to offer. Obviously, the blimp system itself can potentially be upgraded and the details of this improvement will be outlined. Finally, the overall conclusion of this thesis and its application areas will be given.
Chapter 2

Identification and Controller Design

2.1 System Overview

2.1.1 The Mechanical Design of The Blimp

Various problems were experienced while designing mechanical structure of the blimp. Such as for blimp systems, the higher the volume of the envelope, the higher force is needed to drive the system, but a small volume envelope would not be enough to carry the load on the gondola however, a blimp, used indoor, should be sufficiently small. The advantages of using the UAV’s are their ability to stay stationary for a long period of time without consuming energy, additionally, the blimps have low speed and altitude compared to planes and helicopters. Because of the foregoing reasoning’s, the selection of the envelope is crucial for these systems in order to obtain robust tracking of the selected object [3].

The first issue we need to deal is the selection of the size of the envelope which effects the maximum payload that it can carry. Maximum payload is a critical feature that constraints the choice of other onboard components such as the batteries, camera and the balancer bag. The size of the envelope should be big enough to allow us to carry the necessary hardware on the gondola. The second issue is the selection of the batteries. There are two batteries on the gondola. One of which powers the motors and is a 3V battery and the other one is a 9V battery for the camera. Due to the electrical specifications of the wireless camera, constant 9V voltage level is critical.
to minimized the undesired artifacts of captured frame from the camera, the frame quality will decrease with the decreasing voltage. During the test runs, we realized that the 9V battery provided enough energy to operate the camera properly for about one hour. We also tried to find the lightest batteries and the camera possible for the system to reduce the weight of the necessary components in order to increase the mechanical sensitivity of the blimp. Since weight of the gondola will directly affect the size of envelope and the system dynamics. Readers are encouraged to carry out a literature review to make an appropriate selection of the envelope volume, batteries and camera [3, 9].

There are two wireless communication paths involved with the system the first one is due to the wireless communication of the camera and its receiver, and the second wireless communication path is between the motors and the remote controller. Both of these wireless communication paths should have a reasonably long range. If the camera communication has any problems, it will directly affect the camshift tracker, due to the distorted and/or missing frame, and hence the controller algorithm, and will lead to the loss of the robustness of system. The camera and communication tools are the most important components of the system hardware as outlined in figure 3.3 and 3.4 which are used in our system. Additionally if the motors’ communication
path has any problems, the motors will not be able to get control commands from the controller which directly affects the control algorithm.

2.1.2 How the System Works

Initially, once the Matlab code is started, GUI is initiated. Then the camera is started using the GUI. The GUI will start showing the image from the onboard camera. Then the user selects the desired object on the frame which will automatically start the camshift tracker and the tracking box will be shown overlaid on the image. The GUI is explained in Chapter 3 in more detail.

Step by step description of how the system works are given in the next list as well as in figure 2.1.

- Initially, the camera collects the live images and sends the captured image data to camera receiver.
- Camera receiver directly connects to frame grabber card on the computer main board. The captured image is received by the Matlab code.
- After Matlab code gets this image from the frame grabber it is processed with the camshift algorithm and the current location of the tracked object on the frame is outputted. The output of the camshift algorithm is compared to the reference input signal, i.e. the center of the frame, and then error signal is produced. The error signal is then sent to the control algorithm and the output of controller is transferred to the PWM algorithm, which dynamically changes the duration of the on and off cycles of the control pulses.
- The output of the code which is actually output of the PWM algorithm is sent to the parallel port of computer.
- Parallel port is connected to the remote controller transmitter circuit. After transmitter circuit receives the data, it sends it to the motor receiver circuit.
- After the motors receives the command from the transmitter and they drive the blimp according to the desired direction with modulated speed.
- Then, the camera grabs another frame and the same loop is carried out.

After all necessary tools are set up, we can initiate to identify the system.
2.2 Identification

Before applying any control theory tool to any system, we need to know the mathematical model (such as transfer function or the steady state representation) of the system or we need to identify the system to get such a model. In the control theory we assume that the model of the system is known and well-defined. There are two different ways to identify the system. The First one is the experimental method which uses the input output data to get the relationship of the system which is known system identification. The basic procedure is to drive the system by an input signal $u(t)$ and disturbances $v(t)$ and capture the output of the system for the system identification, and then process this data to get a generalizing model of the system. The second way is to identify the system using the Newtonian laws of motion which is the known mathematical modeling of the system [3]. The controller design and identification of the lighter than air vehicles (blimp) is rather difficult because of number of reasons. The first, the blimp should be controlled in three-dimensions, including the control of its altitude. The next difficulty is the payload of the blimp is also limited and the size of the envelope, which directly affects the system model, is also important. The most significant problem for controlling the blimp is its inertia and unstable dynamics which are related to the aerodynamics and the shape. In addition, its dynamics are affected by the local temperature and atmospheric pressure therefore, the dynamics are highly sensitive. So, the controller and pulse width modulation algorithms should be robust enough to deal with this unpredictable dynamics as well as the air flow in the room, which can be considered as the disturbance to the system[10]. Block diagram of the general visual tracking system representation is outlined in figure 2.2. If the blimp dynamics can be properly identified or approximated, then a suitable controller can be designed to track the reference input.

![Block Diagram of Visual Tracking System](image-url)

**Figure 2.2:** Block Diagram of Visual Tracking System.
The Newtonian Laws of Motion are generally used to describe the model of the airships in previous works. These equations consist of gravitational forces, drag forces and thrust/buoyancy forces on the blimp.

The gravitational forces are balanced by the buoyancy forces which are produced by lighter than air vehicle itself. The blimp does not need any energy to maintain its altitude at a certain level, therefore we do not need to take into account for these forces \[10\], since we only consider controlling the blimp horizontally. We will assume that the gravitational and the buoyancy forces are equal to each other and have no effect in the blimp dynamics.

The general aerodynamic drag equation is a practical formula used to calculate the force of drag which is experienced by an object due to movement through the air. \(2.1\) The drag force is directly proportional to the density of the fluid or the mass density \(\rho\). If the density of the fluid or the mass density increases, the drag force will increase. The density of fluid, \(\rho\) is assumed to be fairly small for this setting. The drag force is also directly proportional to the cross sectional area, \(A\) in the direction of the movement. In other words, area refers to the area of contact between the blimp and air which it is moving against. It will be the head of the blimp when the blimp is moving forward and it will be the back side of the blimp when the blimp is moving backwards. The area will be the cross section along the side of blimp when it

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure2.3.png}
\caption{Effect of Aerodynamic Force.}
\end{figure}
is moving right or left. In addition to that, there are other factors affecting the drag force including shape, texture, viscosity, compressibility, lift and the boundary layer separation. In order to represent all these effects on the drag force, coefficient of drag $C$ is also directly proportional to drag force. Furthermore, the aerodynamic drag force is directly proportional to the square of the velocity. For instance, if the velocity increases 2 times, the drag force will increase 4 times. Hence the most important effect is because of the velocity of the blimp. Since actuators of the system will always be saturated by this drag force, it does not matter how fast the propellers thrust the blimp, final speed of the blimp will be constant. The system velocity will be saturated because the aerodynamic drag force will finally overcome the force applied to the blimp by the propellers. See figure 2.3 for a graphical representation of this phenomenon. Once the drag force and the force applied by the propellers are equate the velocity of the blimp is saturated, the blimp will move with constant velocity. We assumed certain parameters to be constant for ease and simplicity of the calculations. We assumed weight, $\rho$ and area $A$ to be very small and that they have no significant effect on the blimp dynamics. On the other hand, even though they have small affect on the blimp dynamics, they will slightly alter the upper and lower limits of saturation. However, the velocity of blimp has significant effect on the blimp dynamics because of the drag force.

$$F = \frac{1}{2} \rho A C v^2 \quad (2.1)$$

The blimp does not have velocity input but it is propelled by thrust instead of velocity [11]. It means we do not have access to velocity state of the system. On the other hand, the propellers give thrust to the blimp. This will also be a challenge for the controller design.

In addition, If a mass system is considered, its differential equation is generally defined as $F = m \ddot{x}$ because of its inertia. The blimp system has also a similar structure, it has relatively small mass with large inertia. There is a force which gives thrust to this mass. So we can define transfer function of the blimp from this differential equation $F = m \ddot{x}$. However, when we are defining this differential equation, we need to keep in mind that this is only the effective part of the blimp dynamics. This is coming from the mass although the transfer function of the generally system has also some dynamics from the DC motors, camera and others tools. Subsequently, we defined this significant part of dynamics of the model $\frac{F(s)}{X(s)} = \frac{1}{\dot{x}_s}$. This is not exactly the transfer function of the general system but we will use this...
dynamics when we are designing the controller. If the controller works practically robust, then we do not need to think about the motor or other dynamics of the equipments. Hereafter, we will define $\frac{F(s)}{X(s)} = \frac{1}{s^2}$ as the transfer function of the system. Furthermore, there should be a saturation block between the two integrators in order to represent the velocity saturation because of the effect of the aerodynamic drag force that we mentioned earlier. The approximate model is in figure 2.4. The limit of the saturation block in the figure 2.4 is defined by the thesis of robust identification of UAV, [9] which is $\pm 1.5$. Nevertheless, these limits can vary due to varying aerodynamic force. It is already explained how the drag force is affected by some parameters in equation (2.1). These limits were tried to be verified by performing some experiments. The plots 2.5, 2.6 will give a better idea about when the velocity is saturated.

![Figure 2.4: Model of The Blimp.](image)

After the approximate mathematical model of the system which contains most important part of the dynamics $\frac{1}{s^2}$, is defined, some experiments are performed to verify the mathematical model. Throughout the process of trying to verify the mathematical model, an appropriate input signal should be chosen. The step input is chosen to be applied in all directions. In addition to this, there is an extra functionality on the GUI for the identification process which is Fw Step to capture the output data to an forward input step. See Chapter 3 for a detailed explanation. The output of the camshift algorithm, which is also the output of the system, outputs two different values which are $x$ and $y$. We know that these are the projected displacements. $y$ value is the horizontal distance between the upper left corner of the image and the red cross sign, showing the center of the object being tracked, while $x$ value is the corresponding similar vertical displacement. Therefore, when the forward step input is applied one needs to look at only $x$ values which are forward displacement. Furthermore, video size is 160X120 (x max 160 and y max 120 pixels) pixels.

Various experiments were carried out and recorded in several occasions in order to compare the real output of the blimp system with the mathematical model of it.
When these experimental data were being collected, the output of the camshift was sometimes losing the selected object for a couple of frames either because of the batteries or wireless communication problems or the loose cable connections. This was resulting in some abrupt changes in the experimental data, therefore, before plotting these datasets, the distorted data or peak values were manually cleaned. Afterwards, cleaned datasets were interpolated by using Matlab spline command to calculate new intermediate data values between existing data values which also helped us to get the uniformly sampled data since the originally captured experimental data is not perfectly uniform sampled because of the computational issues. The Matlab code is attached in Appendix A. After the data were collected and interpolated, the data were plotted as seen in figure 2.5 and 2.6.

Figure 2.5

![Plot of Output of The System-Acceleration-Velocity-Displacement](image)

**Figure 2.5**: Plot of Output of The System-Acceleration-Velocity-Displacement.

In Figures 2.5 and 2.6, the experimental data and curves fitted can be seen. As it can be seen the data plotted in the figures can be very well approximated by a second order polynomial in an interval of \(3 - 11\) and by a linear function after 11 secs. The reasoning behind this issue is the saturation block after the velocity state which can
be seen in Figure 2.4. In other words the system works in linear region of saturation between time intervals 0 to 11. Actually this is very important information about the system. While designing Pulse Width Modulation block after the controller this information in addition to the sampling rate of controller will be useful, see Chapter 4 about detailed analysis of the algorithm development. These results show that the block diagram which summarizes the system dynamics approximates the system properly.

![Figure 2.6](image)

**Figure 2.6:** Plot of Output of The System-Acceleration-Velocity-Displacement.

### 2.3 Controller Design

#### 2.3.1 General Structure of Nonlinear Systems

In the previous section it was shown that the claimed mathematical model \( \frac{1}{s^2} \) dominates the system dynamics in a certain interval, which corresponds to the linear region of the system. Outside of this interval the system acts nonlinearly due to the saturation because of the drag force. If the controller and pulse width modulation
block is designed carefully, the system might be kept in the linear region allowing us to use the simple mathematical model with double integrator.

In the previous section we have shown that the blimp system follows the rules of the block diagram as in Figure 2.4. It is important to note that the saturation block in between the two states which renders the blimp system nonlinearly. In the literature there are some controller design methods for the nonlinear systems usually the nonlinear system is represented as in figure 2.7. If there is an LTI system in the forward path as in figure 2.7, some techniques such as loop transformation to obtain system can be used. If any relay or actuator had a nonlinearity such as saturation or dead zone, it is easy to represent it in the feedback form as in figure 2.7. If the system has the structure of this feedback form, there are some tools to check the stability of the system such as Circle and Popov criteria. These are based on the frequency response of the unforced system. Both Circle and Popov criteria can be used graphically for single-input single-output case. More details can be found in [12], which has profoundly explanation and examples about how Circle and Popov criteria works. The similar type of techniques were also used by the authors of [13, 14].

![Figure 2.7: Feedback Nonlinearity.](image)

### 2.3.2 Controller Design Techniques and Stability Analysis for Nonlinear Systems

Saturation block can be found in different parts of the systems, such as the actuator, the sensor, the controller and
or within the plant itself. Due to some physical and technological constrains, the actuator in the system cannot generally supply enough control action known actuator saturation. Note that the blimp system in consideration has also the actuator saturation in addition to velocity saturation due to the binary operating motors, this is also the reason behind why there should be a pulse width modulation block after the controller block. If the systems have position and rate actuator saturation, Tarbouriech and Garcia showed how to stabilize a system with actuator saturation with full state feedback controller [15]. Tarbouriech et. al. also showed how to design a dynamic output feedback controller when system has actuator and sensor saturation like in figure 2.7. Moreover, They said, if the actuator saturation is not taken care of, a lot of strange things might happen unexpectedly, which suggests us to be extra careful while designing the controller. In addition to the actuator saturation their method is also capable of working under some unmodeled dynamics which are uncertainties [16]. Furthermore there is also some research related with stabilizing of linear systems with actuator saturation, which can also be thought as saturated control. These methods also uses circle and Popov criteria to show the stability region of these kind of systems [14]. If the actuator has saturation or there is an input saturation, there are a lot of different approaches for designing a controller, stabilizing the system, and analyzing of the performance of the system.

The reader is encouraged to see the following references as an example of such works [17, 18, 19, 20]. In addition to these references there are also a lot of papers for saturated systems. Tarbouriech and Garcia applied a different method to design anti-wind up controller for unstable saturated systems especially for aircraft control [21].

However all of these papers show, how to design a feedback controller when the system has actuator saturation, input saturation, and or sensor saturation. Unfortunately, the blimp system has none of these types of non-linearity structures, so the aforementioned methods are not directly applicable, since the blimp has state saturation due to the physics of the problem. Guan and Yang developed some techniques to analyze and design controllers of discrete time linear systems with state saturation. They showed two different analyzing and designing methods for these kind of systems. One of them is applicable when all of states of the system are saturated and the other one is applicable when only some of them are saturated. On the other hand, their techniques also requires full access to the states of the system, which is not the case for the blimp system [22]. Unfortunately we cannot use their techniques for our system because of same reason.
2.3.3 PD Controller Design for The Blimp

For the blimp system, none of the controller design methods in the presented works in previous section cannot be used, since the states of the system is not accessible directly. If the velocity state of the blimp system was accessible, the controller design technique shown by Guan et. al. or one of these controller design methods for nonlinear systems could be used to design a full state feedback controller for the blimp system [22]. Although the blimp system has this limitation, one of these papers was chosen and the developed controller design algorithm was coded to see whether it theoretically works to the blimp system. It worked very well but this feedback controller could not be implemented to the blimp system, so this limitation enforced us to use a different technique to design a controller to the blimp.

Consequently different methods were tried to solve this problem, but the saturation block complicates the controller design procedure, to overcome this issue it was thought, if the propellers make fairly small thrust in order to keep the velocity of the blimp in the linear region of the saturation block, a linear controller design technique could be used. On the other hand, even though a linear controller is designed it is not possible to use it directly since the actuators of the system operates only with two states either full speed forward or backward, which is the second nonlinearity in the system. So before applying the output of the designed linear controller to the actuators it should be post processed with a pulse width modulation block to approximate the overall effect of having speed-controllable actuators. Note that such a PWM approach makes sense since there is an integrator immediately following the controller in effect smoothes out the changes in the input of it to create a smooth speed change of the overall blimp system. See Chapter 4 for a detailed explanation of the PWM. The PWM block can supply very small amount of the power to drive the propellers slowly in order to keep the velocity of the blimp in the linear region of the saturation block. The blimp will be stable with a controller and move pretty slow towards the target object, since its velocity will be forced to stay in the linear region of the saturation block.

Thereafter any digital controller can be designed for double integrator, the approximate model of the blimp, in order to stabilize the system and track the reference input. After designing the digital controller, the PWM block can be added between the system and the controller block. General structure of the whole feedback system can be seen in figure 2.8. The regular controller design techniques such as pole-placement can be used to find a appropriate controller for the double integrator to
stabilize the blimp system and obtain some performance specifications. The reason of considering only double integrator, the most dominant part of the system dynamics, to design the controller is explained in the section 2.2. The closed loop system consisting of the continuous approximate model of the system, the discrete controller, the sampling element, (Zero Order Hold), the saturation block between two integrators and the PWM block are shown in figure 2.8.

Before attempting to design the controller, firstly, the continuous plant should be transformed into its discrete counterpart assuming the sampling element is zero order hold (ZOH) using the equation (2.2). Note that, the continuous plant is type two since it has two integrators, the corresponding discrete time system also has the same property.

From basic linear control system theory knowledge, a stable type two system can track step and ramp inputs perfectly with zero steady state error. So any linear stabilizing controller would achieve the perfect tracking to the objects which are either stationary or moving with constant velocity. Obviously the blimp system cannot be stabilized by the Proportional (P) controller, it is clear that the roots of the system move to the outside of the unit circle which can be seen from the root locus of the system. Obviously trying a Proportional-Integrator (PI) type of controller would not change anything since it only increases the system type but does not bring any additional control action. The other simple two type of controllers are Proportional-Integrator-Derivative (PID) and Proportional-Derivative (PD). The PID type of controller might stabilize the system but degrade the performance of the tracker by creating unnecessary overshoot because of the extra integrator that it brings to the overall system. Note that, since the field of view of the onboard camera is limited excessive overshoot would end up in losing the object being tracked. So the blimp should be moved as slow as possible in order to overcome the issues related with the saturation nonlinearity and video size, but it should be fast enough to track the moving objects. Finally (PD) controller was chosen to obtain zero steady state error and small overshoots since this type of controller does not bring any additional free
integrators to the system. So the PD controller will be considered for the blimp system instead of PID.

\[
G(z) = ZOH \ast \frac{1}{s^3} \\
= \frac{z - 1}{z} \ast Z^{-1}(\frac{1}{s^3}) \\
= \frac{z - 1}{z} \ast \frac{T^2 z \ast (z + 1)}{(z - 1)^3} \\
= \frac{T^2(z + 1)}{2 \ast (z - 1)^2} \\
= \frac{0.18z - 0.18}{(z - 1)^2} \tag{2.2}
\]

\[
D(z) = K \ast (\frac{K_d \ast (z - 1)}{Tz} + K_p) \tag{2.3}
\]

After deciding the controller type shown in equation (2.3), the controller should be implemented for the blimp system and the gains of derivative and proportional controller \(K_d\) and \(K_p\) should also be decided. First of all appropriate ranges for \(K_d\) and \(K_p\) values are needed to be calculated to stabilize the system. Afterwards, the necessary adjustments should be made to obtain the required performance specifications in the real systems.

The bilinear transformation was used in order to be able to use Root Hurwitz for the open loop transfer function of the system and controller together seen in equation (2.5) and (2.4). Then the sampling time is around 0.6secs, note that in reality the sampling time is not perfectly constant because of the varying code running speeds in the computer. See Chapter 4 for a detailed explanation for why the sampling rate is chosen to be 0.6secs. A zero was decided to put close to these two integrator roots in order to make system stable or to bend the root locus curve of the system to inside of the unit circle. To obtain stable system, \(K_p = 0.2\) was chosen.

\[
G(w) = \frac{-0.3w + 1}{w^2} \\
D(w) = K \ast (w + 0.2) \tag{2.4}
\]
In addition, the compensated system was also checked in the $w$ domain with the same PD controller to verify the discrete time performance specifications seen in equation (2.4).

\[
1 + D(z) * G(z) = 1 + \left( \frac{K_d z - K_d + K_p T z}{T z} \right) \left( \frac{T^2 * (z + 1)}{2(z - 1)^2} \right)
= 1 + \left( \frac{K_d T^2 + 2w - K_d + K_p T^2 + 2w}{T^2 + 2w} \right) \left( \frac{T^2 + 2w}{T - 2w} \right) \left( \frac{T^2}{T + 2w - 2w} + 1 \right)
= 16w^2 + 32w^3 + T^5K_p + 4T^3K_dw - 4T^3K_p^2w^2 - 8T^2K_dw^2
\]

\[
C.E = 16w^2 + 32w^3 + T^5K_p + 4T^3K_dw - 4T^3K_p^2w^2 - 8T^2K_dw^2
= 8.1451008K_d - 2.48832K_d^2 - 0.497664 \Rightarrow K_d \Rightarrow 1
= 5.2
\] (2.5)

After defining the characteristic equation by putting $T$ sampling time and $K_p$ values $T = 0.6$ and $K_p = 0.2$ in equation (2.6), Routh-Hurwitz array was used to check to characteristic equation in order to desire the appropriate $K_d$ range while $K_p$ was known. The third row of Routh-Hurwitz array is a second order equation and its roots are $3.2639$ and $0.0623$ which is shown in equation (2.6). This equation is the third element of the first column of Routh-Hurwitz array therefore, it should be positive in order to obtain a stable system. $K_d < 3.263$ should be less than $3.263$ in order to ensure the stability of the blimp. If $K_d$ value is chosen to close to the roots of the second order equation (2.6) or roots of the system close to the unit circle, it causes some oscillation which is not acceptable for the blimp system. In addition, it cannot be less than $0.00623$ $K_d > 0.00623$ therefore, $K_d = 1$ was chosen.

Finally, Jury’s stability test was used to verify the $K_p$ and $K_d$ values achieved the stability of the system. Characteristic equation was already known and $K_d$ and $K_p$ values were checked for the stability analysis by using Jury’s stability test. Discrete time characteristic equation of the system was already defined in equation (2.8) in order to use Jury’s test. Jury’s conditions were checked one by one in order to decide
$K_d$ and $K_p$ values with the purpose of deciding the appropriate controller parameters to stabilize the system and obtain necessary performance specifications.

\[
1 + D(z) * G(z) = 1 + \left( \frac{K_d * z - K_d + K_p * Tz}{Tz} \right) * \left( \frac{T^2 * (z + 1)}{2 * (z - 1)^2} \right)
\]

\[
= 2z^3 - 4z^2 + 2z + T * K_d * z^2 - T * K_d + K_p * T^2 * z^2 + K_p * T^2 * z
\]

(2.7)

\[
Q(1) > 0 \Rightarrow 2 - 4 + 2 + TK_d - TK_d + K_p T^2 + K_p T^2 \Rightarrow 2 * K_p * T^2 > 0 \Rightarrow K_p > 0
\]

\[
(-1)^3 * Q(-1) > 0 \Rightarrow (-1) * (-2 - 4 - 2 + TK_d - TK_d + K_p T^2 - K_p T^2 \Rightarrow 8 > 0
\]

\[
|a_0| < a_n \Rightarrow TK_d < 2
\]

(2.8)

After tested these criteria, any $K_p$ values under the condition $K_p > 0$ can be chosen, hence $K(p) = 0.2$ was preferred. Additionally, choosing of $K_d$ value depends on the T sampling rate, since $T * K_p$ should be less than 2 $TK_p < 2$. T sampling time of the blimp system various around 0.6. See chapter 4 for explanation of the altering sampling time. $K_d = 1$ and $K_p = 0.2$ were chosen. After these calculation processes, the controller were implemented to the system with gain $K = 0.6$ and the approximate model of the system worked pretty well. Jury’s test verified that the calculation of $K_p$ and $K_d$ by using bilinear transformation is also acceptable. See figure 2.9,2.10,2.11 for the root locus of the compensated- uncompensated system and step response of the close loop system. Finally, the K values should be adjusted to obtain necessary performance specs. After the controller was implemented to the system, root locus of the compensated systems were checked in figure 2.10 to decide a reasonable K value. All roots of the characteristic equation were brought on the real axis between 0 and 1 inside of the unit half circle and as far as possible from the unit circle in order to decrease the overshoot. The K value determination is outlined in figure2.10.

Before, the implementation of this controller in the code or closing the loop with negative unit feedback, the time domain version of the controller should be defined which was written with the form in equation (2.9) in order to represent it in time
domain. Next the inverse z-transform of the controller was defined and the input of the controller was known as an error signal seen in equation (2.10). PD controller is physically realizable for the blimp system, since when the controller is running in the closed loop, it needs one step the previous error value. The controller compares the current error value and the previous error value and gives commands according to the differences of the error levels. After the implementation of the controller with the PWM block, the physical system worked very well to track the target object.

\[
D(z) = K \frac{K_d (z - 1)}{T z} + K_p = 1.12 - 1 * z^{-1}
\]  

(2.9)

\[
D(n) = 1.12e[n] - 1 * e[n - 1]
\]  

(2.10)

In addition, some plots were added to show the stability of the uncompensated and the compensated system and the output of the system for the step input. The root locus of the uncompensated, the compensated system and the step response of the system can be seen in figure 2.9,2.10,2.11. Bode and Nyquist plots were also used to check stability of the system and make some performance analysis.

In identification section, some dynamics of actuators, connection paths and some disturbance affects were ignored to calculation simplicity. After deciding the necessary controller gain, the PD controller was implemented to the system which were working properly, however some problems were recognized related to the Pulse Width Modulation algorithm.

The system was working but there is some problems related with pulse width modulation. Equation (2.10) shows controller in time domain, it gets error signal and multiple it with 0.72. Then it gets the previous error signal and multiple it with 0.12. Finally, controller calculates the differences between these two error signal values. Then PWM block gets the output of the controller and decide how much power it needs to apply to the DC motors. However the motors cannot react while the PWM drives the motors under 80 millisecond of duty cycle which is experimentally recognized. The comprehensive explanation of the PWM will also be thoroughly addressed in chapter 4. Due to reaction constrain of the motors, \( K_p \) and \( K_d \) values were tuned to obtain necessary reaction from the motor. In addition, the envelope of the blimp is also rather big which will be explained in chapter 3 for the propellers, thus the motors need to produce more power than the normal case. Finally the
appropriate gain values for controller were decided. $K_p = 2$ and $K_d = 3$ were chosen 2 and 3 which works rather good with this blimp when $K = 1$. Then time domain equation of controller was calculated as follows in equation (2.11).

\[
D(z) = Z^{-1}(7 - 5z^{-1})
\]
\[
D(n) = 7e[n] - 5 \ast e[n - 1]
\] (2.11)

Before going into the next chapter, the steady state error of the system should also be checked. Obviously the system is type two with the PD controller and $e_{ss}$ will be zero as seen in equation (2.12). Actually this zero steady state error is not physically possible for one specific reference point on image with this blimp system however a small frame on image plane was defined as a reference input. Afterwards, the blimp worked almost with zero steady state error.
Chapter 2. Identification and Controller Design

Figure 2.10: Root Locus for the compensated System.

\[
\lim_{z \to 1}(z - 1) * X(z) = \lim_{z \to 1}(z - 1) * \frac{1}{1 + (\frac{K_d z - K_p T_z}{T_z}) * \frac{T^2 z (z+1)}{2 z (z-1)^2}} = 0
\]

(2.12)
Figure 2.11: Step Response of the Compensated System.
Chapter 3

System Structure, Hardware and Software

3.1 System Hardware

3.1.1 Transmitter-Receiver Circuits and Solid State Relay

In order to complete the task of the tracking any stationary or moving object in the room, too many different hardware were needed. Some of the important components of this blimp tracking system are the acquisition, processing, and interpretation of the available camera information and driving the blimp \[23\]. The main hardware are computer, radio controlled transmitter-receiver system to control the propellers on the gondola, the wireless camera and of course the blimp itself. The figure 3.1 shows the modified transmitter which is connected to parallel port used to transmit command signal to propellers of blimp \[9\]. Actually these circuits were a remote controller for a toy (Radio Controlled (RC) blimps) then they were modified to connect to the computer. Only two channels of the transmitter circuit were used to control the propellers since goal of the thesis was control the blimp for forward, backward, left and right except up and down directions. The transmitter circuit on the breadboard gets the signal from the parallel port of the computer and then transmits this signal to the propellers of the blimp. Additionally, three batteries were needed, two of them are 9 V for the transmitter circuit and the camera which is mounted on the gondola, one of them is 3 V for the receiver circuit and the propellers.
Solid state relays were used between the rocket switches and the parallel port output pins; however the parallel port pins were not directly connected to the relays. Resistors were added to the input of the relays in order to limit the current being supplied from the parallel port on the computer as well as to get correct voltage drop for the relay [9]. When one of the pin of the parallel port is high, 5V is sent on the relay output as input to the circuit causing the coil in relay to be energized and closing of the rocket switch. The output of the relays connected directly to the transmitter circuit as shown in figure 3.1 and 3.3. Reasonably small value resistors should be used for the output of the relays since the current is supplied by battery is well enough for the transmitter circuit. The solid state relays used in this project were the part number HSR312 which is shown in figure 3.2 and the series connection schema is also shown in same figure.
3.1.2 Parallel Port

In general, the standard computer parallel port consists of 25 pins seen in figure 3.3. The parallel port pins 2-9 are used as data outputs, 18-25 should be grounded, 10-13 input, 14 auto feed input, 15 error input, 16 initialize and finally 17 selected outputs. 4-7 pins were used for this work as an output. The table 3.1 summarizes the outputs of the parallel port pins and the blimp motion directions. Generally, the parallel port of the computers only supplies small limited currents not enough to control the rocket contacts, therefore the solid state relays were used in order to be supplied much more current to the rockets switches [9]. The figure 3.3 shows the connection of the pins of the parallel port by soldering a wire to the each pin and then running that wire to the corresponding input of the transmitter interface. The transmitter interface took into account whether one or two motors needed to be on. The general information about the RC transmitter-receiver circuit used in this project, the reader is directed to another publication [1]. The figure 3.3 shows the connection of the right turn rocket switch of the first motor to the forth pin of the parallel port via the relay, connection of the left turn rocket switch of the first motor to the seventh pin of the parallel port via the relay, the connection of the right turn rocket switch of the second motor to the fifth pin of the parallel port via the relay and finally the connection of the left turn rocket switch of the first motor.
to the sixth pin of the parallel port via the relay. In addition, the resistor values can be found on the figure 3.3.

![Figure 3.3: Parallel port and Transmitter Circuit Connection.](image)

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>pin(2)</th>
<th>pin(3)</th>
<th>pin(4)</th>
<th>pin(5)</th>
<th>pin(6)</th>
<th>pin(7)</th>
<th>pin(8)</th>
<th>pin(9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward(12)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Backward(48)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Spin Left(24)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Spin Right(36)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Left(8)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Right (4)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.1: Parallel Port Output Pins Control.

### 3.1.3 Blimp-Gondola and Propellers

The UAV’s hardware contained the gondola and propellers commanded by the receiver circuit and powered by the 3 V battery. The gondola consists of the three blades provides the essential force to the motion of the blimp in certain direction. The table 3.1 shows the motion direction of the blimp which are forward, backward, left, right and up, down, however in this research, up and down commands will not be needed to vertically control to the blimp. For instance, if the blimp is turning right, only one motor needed to be on, but if the blimp was to move the forward or the spin left, at that time two propellers needed to be on. Furthermore, the left spin and the right spin commands were used instead of the left and the right commands because of the size of the envelope issue. The envelope of the blimp was replaced
with a larger size envelope, since it was not producing enough lifting force with the regular size envelope to the gondola, camera and batteries. However at that time the propellers stayed same. Therefore only the left or the right commands are not be able to produce enough power to rotate the blimp with an essential speed because these commands run only one blade. After the replacing of the envelope, it caused various problems. Actually the blimp should hover approximately the same altitude the floor. It was going up with new size envelope, then a bag was needed to be added with some ballast on the gondola to keep the blimp in balance in certain altitude. The envelope of the UAV was filled with helium, a non-combustible lighter than air gas. This was done with safety concern in mind [9]. The reader can find more information about how the blimp works [24].

3.1.4 The Wireless Camera and Computer

The camera was intentionally mounted with an approximate angle to the gondola in order to maximize the combination of forward field of view (FOV) and downward field of view (DOV). This gave the UAV to see any object in front of the blimp on the floor. While this angle was being fixed, the system was operated when the camera attached with the different angles, then the best angle was chosen to maximize the FOV and DOV. Actually, this angle is very important in order to increase the system robustness. This wireless camera shown in figure 3.4 operates around the 2.4 GHz and works DC 8V. The receiver of the camera also works DC 9V [9]. The transmission distance is max up to 1200 ft for the wireless camera. During the test process we recognized that the 9V battery provides energy about 1 hour to operate the camera. Aforementioned 9V battery should also be mounted on the bottom of the gondola with a connector between the camera and battery.

2.8 GHz Pentium IV Dell desktop computer was used to run the code. Moreover another important hardware for this research is the frame grabber that is Flash Bus MV Pro. It process the images received by camera. Fortunately Matlab image acquisition toolbox works with this frame grabber. The frame grabber can process around 30 frames/second, however this will not be same with the sampling time of the general system. Flash Bus MV Pro accepts up to eight input triggers so that image acquisition can be synchronized to external events. Processing the images and deciding the controller output in the close loop system will also create same delay for the output of the system. 4/16/15-bit video displayed on the GUI and video in the window screen resolution to 160 x 120. The images by the camera and camshift
frame were showed on the GUI to see the current location of the blimp according to the stationary object. In this way, users can check how the blimp tracks the target object.

3.2 System Software

3.2.1 How to Set Up The Program

First of all, MATLAB were used to process the general blimp system including Graphical User Interface (GUI) seen in figure 3.5. The GUI gives numerous options for the users to control the blimp in close loop mode while the blimp tracks the target object automatically or open loop mode while the blimp is manually controlled with the function of the buttons on the GUI screen.

Prior to run the Matlab code, the users should do various adjustment to their own computers and the computers should also have a frame grabber. The controller commands data couldn’t be sent to the parallel port of the computer by using matlab because of the windows operating system issue. As a result of this problem, a solution for MATLAB was tried to find and then a C++ code was written named pariomat.c to send data for the parallel port. After coding process, it is easy to use matlab to send command for this C++ code. All of the MATLAB and C++ codes were working together smoothly until closing the loop. After closing the loop we recognized that running C++ code by using MATLAB takes too much time (time delay) that we were not able to handle it and then system became unstable.
Chapter 3. *System Hardware and Software*

because of this time delay. Fortunately, MEX-files are a way to call our custom C, C++ or Fortran routines directly from MATLAB as if they were MATLAB built-in functions. Actually this *pariomat.mexw32* file was added in the thesis file but if your file did not come with this *pariomat.mexw32* file, the code probably would not be able to send data to parallel port. Users’s file will have *pariomat.c* and then they can just write *mex pariomat.c* on MATLAB command window. MATLAB will automatically makes necessary *pariomat.mexw32* file in your current browser. After this proccess, the GUI will be ready to automatically or manually control the blimp.

![Graphical User Interface (GUI)](image)

**Figure 3.5:** Graphical User Interface (GUI).

The main goal of the matlab code is to get the image data from the camera by using the frame grabber card, record the current location of the blimp relative to the target object location as a reference and show this color video seen in figure 3.5 on the GUI with the resolution of 160X120 pixels. In other words, if the controller gives a forward command for the blimp, the blimp will move forward. Then location of the blimp will also change according to the target object which is the reference input to track. Meanwhile, the program always gets feedback from the camera.

The matlab code puts new image to the GUI and records the position of the tracked object with respect to the location of the target object in the image in unit of pixel.
Then new location of the blimp is compared with the reference input to decide the error which is the input of the controller. The output of the controller was not directly applied as an input of the blimp because of the physics of the blimp which explained in chapter 2. The PWM block was used between the controller and the approximate model of the system which will also deeply detailed in the algorithm chapter 4. Afterward, according to error signal, the controller decides to the new motion direction of the blimp.

### 3.2.2 How to Use GUI

The user should click the video button to be able to see the video on the video frame of the GUI. Furthermore, the users can stop showed video on the GUI anytime they want by clicking the stop video button. This GUI also has numerous functionalities with the buttons located on left top on the GUI window to manually control the blimp seen in figure 3.5. The user can manually control the blimp for forward, backward, spin left, spin right, left, right direction and stop it by using this named buttons. The user also can check the location of the blimp on the video of the GUI. The user also can immediately stop the motion of the blimp by clicking the stop button anytime in process. In addition to these buttons, one extra Fw Step Experiment button was added with the purpose of giving step input in order to use it for the verification process of the mathematical model of the blimp. After applying the step input for the blimp, \( t_0 \) the time of clicking the Fw Step Experiment button, \( t_{timesave} \) the time for how long the output data was saved which are \( x_{csave} \) and \( y_{csave} \), \( x_{csave} \) x coordinate which is the distance from the left corner of the image to the center of the output of the camshift tracker, \( y_{csave} \) y coordinate which is the distance from the left corner of the image to the center of the output of the camshift tracker are saved with save button. The output of the system will be in unit of the pixel. This Fw Step Experiment was designed for future works for better identification to the system. The save button has functionality to save all essential output information for the better identification. The control buttons are meant to be ON or OFF the motors. The user can change the motion direction of the blimp when motors run. For example when the blimp moves forward, if the user click the backward button, the blimp will move the backward without stopping. The user should also click the stop button in order to turn off the all blades. Using of the GUI is very easy to manually control the blimp. Moreover, before turn off the whole
system, the stop video button should be clicked to prevent the problem related with
the frame grabber.

The objective of the blimp is automatically track a certain target object. The
automatic tracking option on the GUI window was also added to automatically
track the target object. After the user chose the target object by following the step
in the next section, the automatic tracking check box just need to be checked. Then
the blimp will automatically track the target object and try to follow it.

There is same text on the GUI to give various information about the system when
it is processing. Frame rate shows the frame rate of the camera. Confidence level
is also crucial information to decide searching or tracking of the target object. It
shows the probability map of the tracked object by the camshift. If the probability
map of the tracked object is less than decided certain level then blimp searches the
target object. If the camshift tracks the target object, the probability map of the
tracked object will be larger than this chosen certain value. Status text shows that
what the blimp is doing at that time. It can show searching or tracking by looking
the confidence level. Last text is the time sensitivity which shows the sampling time,
in other words, how the system is sensitive. These texts will be deeply explained in
algorithm chapter 4.

3.2.3 How to Choose The Object That Will Be Tracked

(GUI)

Before starting to the automatic tracking, the code should be run to see the GUI.
First of all, the video is started with clicking the video button. Then the user has
various options to choose the target object that the camshift algorithm will track.

The first method of the selection of the target object; when the video is on the GUI,
the user should select any object on the image. However before selecting of the
target object, the select button should be clicked in order to be able to choose any
object on the image by drawing a small square on the target object with the mouse.
Moreover, if the user wants to track a certain object, the object should be in the
field of view of camera to be able to select it. (Initial condition of blimp)

The second method of the selection of the target object; if the user will use same
object to track for his future works, there is a button to save this previously selected
object. The user can save this selected object as a pattern of the target object
any specific file to use later. An object loaded button was also added to load the previously saved pattern object. If the load button is used to introduce the target object to the camshift, the blimp will automatically try to find and track this loaded object with camshift tracker. The user also has chance to save numerous different objects to use them later. It means the user does not need to select same object again and again. Only one time selection of the target object and saving it will be enough for the future works. For example, If the user wants to the blimp to track an umbrella in anywhere in the room, first of all, the user has to locate the umbrella in front of the blimp or initial condition of the umbrella should be in the field of view of the camera. After clicking the video button, the user can choose this umbrella on the video with clicking of the select button by drawing a square with mouse. Then the user can save this selected umbrella by clicking the save button. Now the user can use pattern of the umbrella image to the camshift by using loaded button. After loading the umbrella image, camshift algorithm will automatically track this target object.

The second method of the selection of the target object is very useful functionality of the program or GUI. Aforementioned first method has same disadvantage. When the user uses the first method, the target object should always be located in front of the blimp or in the field of view of the camera. The user has various constrain to use the first method. In the second method, if the user selects the target object earlier and saved it, there is no necessity to put object in front of the blimp. In other words, there is no constraining for the initial condition of the blimp or the target object. Since the program was designed to search the object when the camera does not see the target object. The user just needs to load the pattern of the target object and then blimp automatically starts searching for the target object until finding it. After it finds it then it tries to track it.
Chapter 4

Algorithm

4.1 Camshift Algorithm

4.1.1 General Tracking Algorithm of the System

Object tracking has significant application areas such as robot vision, intelligent traffic systems or monitoring systems. Tracking is a kind of estimation problem of the various target objects in the space by using numerous observation algorithms. In this thesis, the color based tracking camshift algorithm was used which uses a one dimensional histogram to track a specific object with known color tone images sequences. The tracking scenario for this thesis, any specific object can be selected as a target object that is interested for detail analysis. Generally, the camshift algorithm is used as a simplest and easiest way for tracking any colorful object, however it has various problems when it works real time. The significant problem of real time working camshift algorithm is that the output signal of the wireless camera is very noisy, limited accuracy and not fast enough for tracking. In order to overcome these problems, we found a very smart way for the blimp tracking system [2]. Basically, lower level of the mean probability of the observation window was used which is green frame on the image seen in figure 3.5. Then if the current mean probability of the observation window is less than this certain lower level, the general blimp system starts to search for target object until find the certain level of the mean probability. This certain lower level can be decided by checking probability of the target object in camshift algorithm. So, if the camshift loses the target object because of the noise, then it starts to search around the room until find the target.
object, in other words, until obtaining essential mean probability. Matlab was used to code the camshift algorithm. In addition, the image acquisition toolbox of Matlab was also used to get real time image from the wireless camera. Furthermore, this tracking algorithm can be divide three key steps in next itemization.

1. Selection of interesting stationary object in anywhere in the room.

2. Tracking of this target object from frame to frame by using the camshift algorithm.

3. Using the output of the camshift tracking algorithm to control the propellers.

4.1.2 Mean-shift

The basic idea of the mean-shift algorithm is a simple iterative procedure that shifts the each data points towards the highest average of the data points in its neighborhood. The real time mean-shift algorithm endeavors to maximize the correlation between two statistical distributions [25]. The mean shift algorithm will not directly be used as a tracking algorithm, nevertheless the mean shift algorithm is an effective algorithm to work with the camshift tracking algorithm. The Mean-shift algorithm is a non-parametric clustering technique that climbs to find the peak gradient of a probability distribution, in other words, it estimates the maximum density of gradient. It does not need prior knowledge of density of gradient. The mean-shift is a very practical algorithm to decide the modes of the image data sets and define the probability density function in N-dimension space. Right now, following itemization will give details about how the mean-shift algorithm works [2].

- Set a search window size on image plane.

- Choose initial position of search window.

- Calculate center of mass inside of the search window.

- Move the center of search window to the center of the mean location computed previous step.

- Repeat step-3 and step-4 until converge the mean value that computed step-3.
The image is a collection of data and the differences or properties of objects can be observed by looking this image dataset. Besides the mean shift algorithm is applied on this data set or image data to find the peak density of the gradient. The main disadvantage of the Mean-shift algorithm for the blimp system is that the mean-shift algorithm works for the static probability density function. However, the blimp system is commanded according to real time image data. It means the probability density function of the real time data can change while blimp is moving towards to the target object. Additionally, if the tracked object size changes on the image when blimp is moving forward or backward, this changing size of the object also will be a problem for mean-shift algorithm, since it does not have ability to resize the search window according to tracked object size.

Due to the disadvantage of the mean-shift algorithm for real time tracking, the camshift algorithm was applied to the blimp application. Since the camshift tracking algorithm contains also mean-shift algorithm. The main differences between mean-shift and camshift are that the search area of the camshift algorithm is little bigger than mean-shift algorithm and the search window size is also automatically adjusted with the zero moment by the camshift algorithm. That’s why the setting of the search window size is a crucial process for mean-shift and the camshift algorithm.

4.1.3 Camshift

The camshift algorithm was used to object tracking which is widely used in computer vision applications especially for the color space. The camshift algorithm, or Continuously Adaptive Mean-Shift algorithm, is a modified version of the Mean-shift algorithm, developed by Intel [26]. The camshift uses color based histogram to define characteristics of the target object and mean-shift algorithm to search the image. Bradski modified the mean-shift algorithm to cope with dynamically changing of the color probability distributions derived from image sequences [26]. In other words, each single image frame is changed to color probability distribution function regarding to the histogram of the target object. Moreover, the camshift algorithm decides the center and size of this selected target object. It was actually developed as a fast and efficient way to track a given face in the presence of noise then it used to track other colorful objects as well.

In order to track a target object by using camshift, a probability distribution image of the desired color in the video must be crated [26]. Camshift initially process
from RGB color system to HSV, (hue is the color type (such as red, black, or green), saturation is the "vibrancy" or "purity" of color, and value is the brightness of color) color system. Additionally, the camshift algorithm separates the Hue from the Saturation and Value in order to create 1D histogram from the Hue channel. Besides, the model histogram is used to covert the incoming frames into probability image.

The search window is automatically adjusted during searching process. However, the initial window size should be any reasonably arbitrary value. The size is a function of the zero moment which presents the distribution area under the search window. In other words, the search window radius, or height and width, is set to a function of the zeroth moment during search. The search window of the camshift is bigger than the mean-shift window and the search window of the camshift is also adjusted by algorithm during search of target object [26, 27].

The basic algorithm of the camshift will be represented in next itemization and also in figure 4.1, [26].

1. Choose the initial location of the search window.
2. Mean-shift as above (one or many iterations); store the zeroth moment.
3. Set the search window size equal to a function of the zeroth moment found in step 2.
4. Repeat steps 2 and 3 until convergence (mean location moves less than a pre-set threshold).

As long as the separate objects in the image do not have same properties and the image data of the camera is not noisy, the camshift tracker works sufficiently good for the blimp system to track a target object. However, if there are two same shape and color objects in the image and the image data is slightly noisy, the camshift algorithm probably cannot track to the selected target object properly. It can jump one to other. These kinds of problems are not acceptable for the blimp tracking system, since the blimp system is very sensitive, the reason of the sensitivity was explained in chapter 3 and 4. The general algorithm of the system cannot overcome that kinds of disturbances to keep the stability of the system. Camshift algorithm has to follow the target object without even one time loosing in order to keep the stability of the blimp. This sensitivity is not directly related with the camshift, it
is also related with the hardware selection of the system. In this work, propellers are fairly powerless to make a thrust the blimp. All combination of these problems makes the system very sensitive.

Figure 4.1: Block Diagram of Color Object Tracking.

The camshift algorithm of the system was added in Appendix B and each function of the camshift algorithm was numerated. The code in Appendix B tries to track an object by finding regions that best match the characteristics of the target object.
4.2 Controller and PWM Algorithms

Pulse width modulation is a kind of method for reducing the amount of power delivered to a DC motor. It is a kind of averaging the power delivered to motor. Instead of reducing the voltage to the motor, the power supply of the motor is rapidly switched on and off. If the voltage is reduced, it will automatically reduce the power operating the DC motor. The term duty cycle describes the proportion of on time to the regular interval or period of time; a low duty cycle corresponds to low power or off time, since the power is off for most of the time. Duty cycle is expressed in percent, 100 being fully forced. Additionally, The PWM is crucial part of the thesis in order to overcome the nonlinearity of the system.

In this research, after choosing the target object which is anywhere on the image, the target object will be tried to be brought to center of the image. At the beginning of the process to successfully complete the task of tracking any object in the image, the reference input of the system which is a small frame in the center of the image needs to be decided in order to close the loop with visual feedback and the controller explained in the chapter 2. It is obvious that reference input should be any pixel values for X and Y axis of image to define the center of image. Due to the problems explained in Chapter 2 and 3 such as big envelope for the propellers, sensitivity and noisy image, the reference input cannot be a single point. It is almost impossible to stabilize the system on this single point with this controller and these hardware. If a single point was chosen, the blimp would always oscillate around this point, however it would never stop because of its great inertia. So a small frame was defined as a reference input which is navy blue on the video seen in figure 3.5. The coordinates of the reference input are also in the table 4.1 which exactly represent a frame in center of the video. After deciding the reference input, the controller shown in equation 4.1 need to be added in the loop.

<table>
<thead>
<tr>
<th>Axis</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xrefmin</td>
<td>40</td>
<td>Y</td>
</tr>
<tr>
<td>Xrefmax</td>
<td>80</td>
<td>Y</td>
</tr>
<tr>
<td>Yrefmin</td>
<td>X</td>
<td>60</td>
</tr>
<tr>
<td>Yrefmax</td>
<td>X</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 4.1: Reference Input.

First of all, the current position of the blimp has to be decided. This is crucial information to decide driving direction of the blimp. The differences between reference input seen in the table 4.1 and current position of the target object is defined
as error signal in code. According to this error value, the driving direction of the blimp is chosen. In other words, according to results of this comparison, blimp can be driven right-left or forward-backward. For instance, if the current position of the target object is close to the center of X axis and far from the Y axis, it means that the blimp should be made a thrust to right or left in order to centralize Y axis. If it is close to the center of Y axis and far from the X axis, the blimp should be made a thrust forward or backward in order to centralize X axis. After deciding right-left or forward-backward pairs, the algorithm chooses one of the right or left pairs and one of the forward or backward pairs to decide the motion direction of the blimp seen in figure 4.4.

\[
D(n) = 7e[n] - 5 \times e[n - 1]
\] (4.1)

After deciding the motion direction of the blimp, error signal goes through the controller. Controller gets this current error signal and multiple it by 7, then the controller gets previous recorded error signal and multiple it by 5. Finally, the controller subtracts these two values. This critical information of the controller output is used to decide the percentage level of the PWM or the average value of power to operate the propellers. Before going through the PWM block, the numerous parameters of the PWM have to be decided. In other words, the algorithm calculates that how much propellers need to make thrust to blimp according to current position of target object. Additionally, one larger frame was experimentally set around the reference frame. All boundaries of the larger frame around the reference frame can be seen in table 4.2. Furthermore, two figures was plotted to show how the PWM works to increase robustness of system seen in figure 4.2 and 4.3. Since when the blimp moves towards to the reference frame, the blimp has to be made thrust with as small as possible power according to closeness to the reference frame. Its inertia is used by reducing the power of propellers until it reaches the reference frame.

<table>
<thead>
<tr>
<th>Forward-Backward</th>
<th>P. PWM*100</th>
<th>Right-Left</th>
<th>P. PWM*100</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X &lt; 20 )</td>
<td>1</td>
<td>( Y &lt; 30 )</td>
<td>1</td>
</tr>
<tr>
<td>( 20 &lt; X &lt; 40 )</td>
<td>( \text{Cont.out}/\text{Maxpwm}2 )</td>
<td>( 30 &lt; Y &lt; 60 )</td>
<td>( \text{Cont.out}/\text{Maxpwm} )</td>
</tr>
<tr>
<td>( 40 &lt; X &lt; 80 )</td>
<td>0</td>
<td>( 60 &lt; Y &lt; 100 )</td>
<td>0</td>
</tr>
<tr>
<td>( 80 &lt; X &lt; 100 )</td>
<td>( \text{Cont.out}/\text{Maxpwm}2 )</td>
<td>( 100 &lt; Y &lt; 130 )</td>
<td>( \text{Cont.out}/\text{Maxpwm} )</td>
</tr>
<tr>
<td>( 100 &lt; X &lt; 120 )</td>
<td>1</td>
<td>( 130 &lt; Y &lt; 160 )</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.2: Percentage level of PWM.
If the target object is outside of the large frame, the blimp has to be made a thrust with full force, in other words PWM block should produce one hundred percent power so as overcome the problems related with big envelope with small propellers and inertia of the blimp. For instance, if it is not pushed with full force against the its moving direction when it is around the boundaries of video screen, the blimp can lose the target object on the screen. Since if enough force was not applied to the blimp in order to stop it inside the region of the screen, it would keep moving due to its inertia. Then system becomes unstable because of overshoot. That’s why the blimp is pushed with full force towards to center when it is around the boundaries of the image. After we manage to keep the target object to inside of the image plane or stop the blimp, we have to bring the target object to the center of reference frame.

On the other hand, if PWM block is not added in the system, the propellers always push the blimp with full force and most probably the blimp can not stop the inside of the reference frame due to its great inertia. The propellers have to be stopped to make thrust the blimp when the target object is in the reference frame. However it will keep moving with its inertia towards the other side of the reference frame. Moreover, the propellers push the blimp to the opposite direction according to previous thrust direction in order to again bring it to the center. If the propellers again push the blimp with full force, same situation will again happen. Finally, the blimp will never stop in this reference frame. It will always oscillate around the reference frame without stopping. This situation happens for both direction forward-backward or right-left. To solve this problem, the propellers has to push the blimp with as small as possible force when the blimp comes toward to the center of the reference frame. The figures 4.2 and 4.3 show how the PWM reduces the power of propellers with the purpose of overcoming the inertia.

Furthermore, various parameters of the PWM algorithm needed to be decided. Slot number represents that how many times the blimp needs to be made thrust with full force to obtain essential average power for each cycle or in one period. Two and one slot was tried. For instance, if two slots is used, the power operating the motor needs to be divided by two or the blimp need to be made thrust two times with full force in one period. It means that the operating power of the motors is delivered with two cycles in one period. The period is representation of the sampling time of the controller. However, we experimentally recognized that the DC motors cannot give reaction if operating duty cycle of the PWM for the motors is less than 80 millisecond explained in chapter 2. If the blimp was made thrust with full force below 80 millisecond, motors could not work with this amount of the delivered power.
Chapter 4. Algorithm

Figure 4.2: Percentage level of PWMX.

Probably each duty time of each cycle will be less than 80 millisecond, if the slot number is chosen two. Afterwards, the slot number was chosen one to solve this problem.

Moreover, the blimp system is saturated around 0-11 second which is already explained in section 2.2. The saturation time of the system is also fundamental information to decide maximum time of each slot or sampling time of the controller. The period of system is around 0.6 second. This sampling time is also reasonable for the PWM algorithm. When the sampling time of the system or period of the PWM is chosen, the reaction level of motors and saturation time of the blimp have to be considered together. The period of the PWM or the sampling time of the controller should be between the reaction level of motors and saturation time of the blimp. On the other hand, the sampling time should be as fast as possible, since the field of view of the camera is fairly small and it can lose the target object when the blimp is operating. The algorithm should decide quickly and then drive the system according to commands. Then the algorithm should get new image to decide new commands to drive the system, otherwise there can be overshoot in the sampling time. Hence overshoot makes system unstable.
After deciding the slot number, maximum level of the PWM also has to be decided. Two different values were defined for right-left and forward-backward directions which are maxpwm and maxpwm2 in the code. These numbers was used to decide the percentage level or proportion of the PWM. They are seen in graphics 4.2, 4.3 and also on table 4.2. These numbers are crucial to prevent overshoot and make the system more robust. The block diagram of the general algorithm was seen in figure 4.4.

The block diagram of the general system is represented in figure 4.4 which shows how the closed loop system works. There are still some problems related with reaction level of the motors which is around 80 millisecond. It happens when the target object pretty closed to reference frame. At that time, duty cycle time could be less or equal 80 millisecond and PWM applies some power to the propellers but they are not be able to work with that amount of the power. Actually this is not a big problem for this blimp problem because when blimp is moving towards the reference frame, it keeps moving because of its inertia even though propellers stop.

This PD controller, the PWM and the camshift algorithms are the most important part of the system. That’s why we have to be careful during the combination process
of these three algorithms and deciding the parameters of the PWM algorithm. Every single of these algorithms should work consistent with the others. If any problems occurred one of them, it would directly affect the system stability. For instance, the blimp system has a problem with camera connection, sometimes camshift algorithm cannot get image or gets some distorted image from the camera. This distorted image can break the camshift algorithm and indirectly the PWM and the controller algorithms also. The general code of the system was added in Appendix C.
Chapter 5

Conclusion and Future Works

The purpose of this study was tended toward applied research. This thesis demonstrated an effective way to track any defined object with unmanned aerial vehicle or lighter than air vehicle by using camshift algorithm, PD controller and the Pulse Width modulation. The blimp control software was also developed. First of all, we tried to decide a suitable mathematical model or the transfer function for this unknown blimp system. The identification process of the blimp is very hard since it has rapidly changing dynamics which consists of DC motors dynamics, blimp itself dynamics and disturbances. Moreover, the blimp is also hard to control because of its great inertial force and air resistance. In addition, it has nonlinearity velocity saturation because of the aerodynamic force (air resistance. A PD controller was designed with Pulse Width Modulation for the blimp as an indoor application. By using the PWM block and PD controller block, a suitable control structure was improved for the blimp system. Actually, the PWM algorithm was added in the loop in order to handle saturation nonlinearity. Finally, The algorithms including the PWM, the camshift and the PD controller have been successfully improved to track the target object. In addition, a software was developed which has very basic and useful interface for users seen in figure 3.5. Users can select any object in the room and give command to the blimp to find and track it. Users also can manually control the blimp by using this GUI. Future work for the general system should be focused on a few different areas. The first area should focus on making the current system more robust.

For the identification process, input/output relations were used, one can developed a suitable model of LTI system, and the Newtonian Laws of Motion were also used, one can developed the mathematical model of the blimps, to use during the controller
design process. We recognized that the blimp system is not exactly LTI during the identification process by using input/output relations. Then we verified this reality by comparing with mathematical model, since it has nonlinearity in velocity state. The upper and lower levels of saturation block were also decided in the loop during identification process. However, for future work, we used approximated mathematical model of the blimp or significant part of the dynamics as a transfer function to use during the controller design process. It was fairly poor. If we had more uniform dynamics or transfer function of blimp, we could design most robust system. In this case, identification process should run over to find more accurate transfer function. For instance, we did not consider the mass and some parameters of aerodynamic force such as mass density $\rho$, cross section area ($A$), they directly affect the saturation block or system dynamics. The DC motors dynamics should also be added to dynamics of general system for future works. In addition, the air condition of the room also effects the system dynamics, so during the identification process of the system, it can be represented as an uncertainty. Moreover, if re-identification is needed, there is some problem related with the motors. One of them has a problem and cannot give same force the other one up. That’s why they should be replaced with new ones and exactly same motors. Otherwise, it will also change the general structure of system dynamics.

The system integration part of the thesis involved gathering the real time images captured by the wireless camera to perform motion tracking using the camshift algorithm and sending the data of the tracked feature to the frame grabber and also controller and PWM algorithm. Then the PWM and controller algorithms command the blimp with an appropriate control action so as to obtain that the blimp hovers continuously over the target object. However, the blimp system has various problems during gathering the real time video images captured by wireless camera. Sometimes the camshift algorithm cannot track the target object because of noisy images. Since using camshift and real time image data simultaneously is not perfect. For future works, some hardware part of the system such as camera, transmitter and receiver circuits and camera receiver can be replaced with more quality ones. For instance, connection between the battery and the camera can make some distortion for the image data and this distorted data affects the camshift algorithm. Then these distorted images indirectly make system unstable. The algorithm that we developed can overcome some small distortion on captured images but camshift algorithm can be advanced by adding some filters such as Kalman to improve its robustness for future works. The camshift algorithm is the beginning of the general
algorithm, explained in section 4.4, that’s why any tiny problem can totally effect system stability.

The next stage of the research is controller design for approximate mathematical model of the system. Basic linear control system theory is used to design PD controller by looking root locus of the system 2.9. Designed controller can be seen in equation 2.10. However implementation process of the designed controller causes some problems for the real system. Then the $K_p$ and $K_d$ values were tuned to obtain essential performance criteria. This tuning process was necessary to compensate the unmodeled dynamics of the system. Finally, the Pulse Width Modulation block was added between model of the system and designed controller block to solve nonlinearity problem. The blimp velocity was kept in the linear region of the saturation block by the PWM block. Additionally, numerous different methods can be developed to handle this kind of nonlinearity. For instance, after defining the uncertainty of system in order to represent the nonlinearity, robust controller design techniques can be used to developed robust controller in order to improve the robustness of the current system. In addition, the PWM parameters such as slot number or maxpwm level can be improved to design more robust system. Since system performance can be advanced by new controller and more efficient PWM algorithm. The motors can be replaced from ON-OFF to a proportional type system. Again before changing the algorithms and controller, better identification is necessary and some hardware of the system also should be changed.

This entire work was academically inspiring, enjoyable and it gave me a lot of experience related with application of parallel port, camshift algorithm, controller design and PWM. This is a cheap, efficient technology which can be put to use advertising, aerial photography and intelligent traffic systems. We set up the entire system for under 200 dollars. "This system could be used for traffic control, to observe and predict traffic rush hours and take pre-emptive measures to avoid these. It could be used to fight forest fires where a blimp fitted with either a camera or thermal sensors or a combination of both can survey affected areas and facilitate in the combat against these. Other possible applications are atmospheric studies, study of irrigated crop land to survey water supply distribution, etc.,”[1]
Appendix A

Interpolation of Experimental Data

function [yy tdesired u]=interpolation(timesave,xcsave);
tdesired=linspace(timesave(1),timesave(end),length(timesave));
yy=spline(timesave,xcsave,tdesired);
tt=tdesired;
u=ones(size(yy));
figure;plot(timesave,xcsave,'b.');
title('Plot of Output of The System-Acceleration-Velocity-Displacement');
xlabel(' Saving Time of Data');
ylabel('Output of The System');
hold on; plot(tdesired,yy,'r+');
tt(:,end-1)=[];
plot(tt,diff(xcsave),'r-');
tt(:,end-1)=[];
plot(tt,diff(diff(xcsave)),'g*');
h = legend('Real Displacement','Interpolated Displacement','Velocity','Acceleration',2);
for x = 4:.01:17;
y = 0.6*x^2-4;
hold on; plot(x,y,'b-')
z =6*x-6;
plot(x,z,'y.')
end
Appendix B

Camshift and Mean-Shift Algorithm

1- function w = MeanShift(map, w, iter, minmov)
% MeanShift(map, w, iter, minmov)
% This function climbs the gradient of a 2D probability map
% map - a probability map
% w - the meanshift window
% iter - maximum number of iterations
% minmov - minimum movement accepted
xc = round(w(1) + (w(2)-w(1))/2);
yc = round(w(3) + (w(4)-w(3))/2);
for j=1:iter
    m00 = Moment(0,0,w, map(:,:));
    if (m00 <= 10)
        break;
    end
    m01 = Moment(0,1,w, map(:,:));
    m10 = Moment(1,0,w, map(:,:));
    oldx = xc;
    oldy = yc;
    xc = m10/m00;
    yc = m01/m00;
    xmov = round(xc - oldx);
    ymov = round(yc - oldy);
w = [(w(1)+xmov) (w(2)+xmov) (w(3)+ymov) (w(4)+ymov)];
if (norm([oldx oldy] - [xc yc]) < minmov)
  break;
end
end

function w = CamShift(map, w, iter, minmov, boxresize, ratio)
  xc = round(w(1) + (w(2)-w(1))/2);
  yc = round(w(3) + (w(4)-w(3))/2);
  oldw=w;
  for i=1:iter
    oldx = xc;
    oldy = yc;
    w = meanshift(map, w, 10,1);
    xc = round(w(1) + (w(2)-w(1))/2);
    yc = round(w(3) + (w(4)-w(3))/2);
    m00 = Moment(0,0,w,map);
    if (m00 <= 10)
      xinc =round((w(2)-w(1))/4);
      yinc =round((w(4)-w(3))/4);
      w = [(w(1)-xinc) (w(2)+ xinc) (w(3)-yinc) (w(4)+yinc)];
      continue;
    end
    wh2 = round(sqrt((m00)/ratio)/2*boxresize);
    ww2=round(wh2*ratio);
    w = [(xc-ww2) (xc+ww2) (yc-wh2) (yc+wh2)];
  end
  oldw = w;
end

% CamShift(map, w, iter, minmov)
% This function climbs the gradient of a 2D probability map
% map - a probability map
% w - the initial Camshift window
% iter - maximum number of iterations
% minmov - minimum movement accepted
% xc = round(w(1) + (w(2)-w(1))/2);
% yc = round(w(3) + (w(4)-w(3))/2);
% m00 = Moment(0,0,w,map);
%ShowIntermediateResults(map, rgbframe, w, hmap, htrack);
% s=1.1*sqrt(m00);
% w=round([xc-ratio*s/2 xc+ratio*s/2 yc-s/2 yc+s/2]);
diff= abs(oldw-w);
if (sum(diff(:)) < minmov)
    break;
end
w=oldw;

3- function h = BuildHistogram( region, tls, ths, tlv, thv, component, bins )
%BuildHiistogram(region)
% Builds the histogram ignoring pixels with low values for saturation
% and value
[sx sy c] = size(region);
h = zeros(bins,1);
for i=1:sx
    for j=1:sy
        if (region(i,j,2) > tls) && (region(i,j,2) < ths) && ...
(region(i,j,3) > tlv) && (region(i,j,3) < thv)
            idx = round(region(i,j,component)*(bins-1)) + 1;
            h(idx) = h(idx)+1;
        end
    end
end

4- function h = AdjustHistogram(h, factor)
h = h + factor;
for i=1:length(h)
    if (h(i)>1)
        h(i)=1;
    elseif (h(i)<0)
        h(i)=0;
    end
end

5- function imageupd(hObject, eventdata,handles,vid)
data = peekdata(vid,1);
axes(handles.axes1);
gca
handles.axes1
h=imshow(data,[ ]); drawnow;
h
flushdata(vid);

6- function m = Moment(i,j,w, I)
% Moment(i,j,w,I) - computes the ith, jth moment from the image
% within window w
m = 0;
[sx sy] =size(I);
for h=max(w(1),1):min(w(2),sx)
  for k=max(w(3),1):min(w(4),sy)
    m=m+I(h,k)*h^i*k^j;
  end
end

7- function img = ProbabilityMap2( frame, histt, tls, ths, tlv, thv, ...component, bins )
%ProbabilityMap(image, hist, tls, ths, tlv, thv, component, bins )
% Build the probability map of a frame using hist.
[sx sy c] = size(frame);
%n = sum(hist(:));
img=zeros(sx,sy);
for i=1:sx
  for j=1:sy
    img(i,j)=histt(round(frame(i,j)*(bins-1) + 1));
  end
end

8- function h = Gausshistogram(h,sigma)
[x idx]=sort(h);
[sx sy] =size(h);
gauss = FSPECIAL('gaussian',sx,sigma);
gauss= gauss(:,round(sx/2));
shiftvalue = idx(sx) - round(sx/2);
h = circshift(gauss,shiftvalue)

9- function w=rec2w(rec)
w=round([rec(2) rec(2)+rec(4) rec(1), rec(1)+rec(3)]);

10- function rec=w2rec(w);
rec=round([w(3) w(1) w(4)-w(3) w(2)-w(1)]);
Appendix C

Remote Control

function varargout = remotecontrol(varargin)
warning off;
% REMOTECONTROL M-file for remotecontrol.fig
% REMOTECONTROL, by itself, creates a new REMOTECONTROL or raises
% the existing singleton*.
%H = REMOTECONTROL returns the handle to a new REMOTECONTROL or
%the handle to the existing singleton*.
%REMOTECONTROL('CALLBACK',hObject,eventData,handles,...) calls the
%local function named CALLBACK in REMOTECONTROL.M with the given input arguments.
%REMOTECONTROL('Property','Value',...) creates a new REMOTECONTROL or raises the
%existing singleton*. Starting from the left, property value pairs are
%applied to the GUI before remotecontrol_OpeningFcn gets called. An
%unrecognized property name or invalid value makes property application
%stop. All inputs are passed to remotecontrol_OpeningFcn via varargin.
%*See GUI Options on GUIDE’s Tools menu. Choose "GUI allows only one
%instance to run (singleton)".
%See also: GUIDE, GUIDATA, GUIDATA
% Edit the above text to modify the response to help remotecontrol
% Last Modified by GUIDE v2.5 25-Mar-2010 18:42:12
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @remotecontrol_OpeningFcn, ...
'gui_OutputFcn', @remotecontrol_OutputFcn, ...
'gui_LayoutFcn', [], ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

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

% UIWAIT makes remotecontrol wait for user response (see UIRESUME)
% uiwait(handles.figure1);

global rect_selected;
global t0;
t0=0;
rect_selected=0;

% --- Outputs from this function are returned to the command line.
function varargout = remotecontrol_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
Appendix C. Remote Control

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

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

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

vid = videoinput('winvideo',1,'RGB24_160x120');
set(vid,'TriggerRepeat',Inf);
vid.FrameGrabInterval = 1;

start(vid);

global done;
global rect_selected;
global objwind;
global data;
global h;
global yc;
global track;
global abstime;
global t0;
global confidence;
refr=1;

t0=0;
last_controller_access=-1;
last_cont_samp=-1;
done = false;
track=false;
seterrors1=0;
seterrors2=0;
axes(handles.axes1);
oncycle=1;
counttt=1;
while ~done

    if rect_selected
        st=tic;
        xcrefmin=40;
        ycrefmin=60;
        xcrefmax=80;
        ycrefmax=100;
        data = getdata(vid,1);
        if(~isempty(data))
            if(refr)
                datawr=data;
                datawr(xcrefmin:xcrefmax,ycrefmin:ycrefmax,1)=0;
                datawr(xcrefmin:xcrefmax,ycrefmin:ycrefmax,2)=0;
                image(datawr); %drawnow;
                axis off;
            end
            frame=rgb2hsv(data);
            tsl=0.05;
            tsh=2;
            tbl=0.05;
            tbh=0.95;
            bins=32;
            boxresize=1.6;
            component=1;
            w=rec2w(objwind);
            ratio= (w(2)-w(1))/(w(4)-w(3));

            map=ProbabilityMap(frame, h, tsl, tsh, tbl,tbh, component, bins);
            w=rec2w(objwind);
            w = CamShift(map, w, 10,1,boxresize, ratio);
            objwind=w2rec(w);
            xc = round(w(1) + (w(2)-w(1))/2);
yc = round(w(3) + (w(4)-w(3))/2);
%% calculate the current confidence
try
    current_confidence=mean(mean(map(max(1, objwind(2)):
min(120, objwind(2)+objwind(4)),
max(1, objwind(1)):min(160, objwind(1)+objwind(3)))));
catch
    stop(vid);
end
set(handles.text4,'String',
['Confidence= ',num2str(abs(current_confidence-confidence))]);
if(refr)
    rectangle('Position', objwind,'EdgeColor',[0 1 0]);
    hold on;
    plot(yc,xc,'r+','LineWidth',1); % drawnow;
    hold off;
    % controller gets this information
    if(get(handles.checkbox1,'Value')==1)
        pwmtime=0.6;
        slots=1;
        slottime=pwmtime/slots;
        controller_access=toc(abstime);
        counttt=counttt+1
        % decide if we need new decision
        time_passed=controller_access-last_controller_access;
        cont_samp=controller_access-last_cont_samp;
        set(handles.text3,'String',
            ['Time sensitivity= ',num2str(round(cont_samp*1000))]);
        last_cont_samp=controller_access;
        if((time_passed-pwmtime)>0)
            %counttt=1
            last_controller_access=controller_access
        setdif=[ycrefmin:ycrefmax]-yc;
        [abser idx]=min(abs(setdif));
        seterror1=sign(setdif(idx))*abser;
        setdif2=[xcrefmin:xcrefmax]-xc;
[abser2 idx2]=min(abs(setdif2));
seterror2=sign(setdif2(idx2))*abser2;
%seterror1 = max(abs(xc),xcrefmax-xc)
seterrors2=[seterrors2 seterror2];
seterrors1=[seterrors1 seterror1];
controllerout=7*seterrors1(end)-5*seterrors1(end-1);
controllerout2=7*seterrors2(end)-5*seterrors2(end-1);
%%if positive turn right negative left
maxpwm=12*10;
maxpwm2=12*5;
%on percentage of motors for right left action
perc=sign(controllerout/maxpwm)*min(1,abs(controllerout/maxpwm));
perc2=sign(controllerout2/maxpwm2)*min(1,abs(controllerout2/maxpwm2));
if abs(current_confidence-confidence)<1
    if(abs(perc)>abs(perc2))
        if(perc<0)
            ontime=((slottime*abs(perc)));  % ontime in seconds
            offtime=((slottime-ontime));
            parcode=36;
        elseif(perc>0)
            ontime=((slottime*abs(perc)));  % ontime in seconds
            offtime=((slottime-ontime));
            parcode=24;
        else
            ontime=((slottime*abs(perc2)));  % ontime in seconds
            offtime=((slottime-ontime));
            parcode=0;
        end
    else
        if(perc2>0)
            ontime=((slottime*abs(perc2)));  % ontime in seconds
            offtime=((slottime-ontime));
            parcode=12;
        elseif(perc2<0)
            ontime=((slottime*abs(perc2)));  % ontime in seconds
            offtime=((slottime-ontime));
            parcode=48;
        end
    end
else
    if(perc2>0)
        ontime=((slottime*abs(perc2)));  % ontime in seconds
        offtime=((slottime-ontime));
        parcode=12;
    elseif(perc2<0)
        ontime=((slottime*abs(perc2)));  % ontime in seconds
        offtime=((slottime-ontime));
        parcode=48;
    end
end
else
  ontime=((slottime*abs(perc2)));
  offtime=((slottime-ontime));
  parcode=0;
end
end
else
  perc=0.75;
  ontime=((slottime*abs(perc)));
  offtime=((slottime-ontime));
  parcode=24;
end
pariomat(parcode);
last_on_active=toc(abstime);
end
perc
perc2
if(oncycle)
  slotnumber
  active=toc(abstime);
  time_passed=active-last_on_active;
  if(time_passed<ontime)
    pariomat(parcode);
%last_on_active=toc(abstime);
  else
    pariomat(0);
    last_off_active=toc(abstime);
    oncycle=0;
  end
end
if(~oncycle)
  active=toc(abstime);
  time_passed=active-last_off_active;
%last_off_active
  if(time_passed<offtime)
    pariomat(0);
%last_off_active=toc(abstime);
else
    slotnumber=slotnumber+1;
    if slotnumber<slots
        pariomat(parcode);
        last_on_active=toc(abstime);
    end
    oncycle=1;
    end
end
end

et=toc(st);
else
    st=tic;
    data = getdata(vid,1);
    if(~isempty(data))
        image(data); %drawnow;
        axis off;
    end
    et=toc(st);
end
fr=round(1/(et));
set(handles.text1,'String',['frame rate= ',num2str(fr)]);
flushdata(vid);
end
stop(vid);
delete(vid);
clear vid;
rect_selected=0;

% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
global done;
global track;
done = true;
track=true;
% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
global rect_selected;
global objwind;
global data;
global h;
global abstime;
global confidence;

% hObject handle to pushbutton12 (see GCBO)
% handles structure with handles and user data (see GUIDATA)
rect_selected=1;
rgbframe=im2double(data);
frame=rgb2hsv(rgbframe);
reg = frame(objwind(2):objwind(2)+objwind(4),
objwind(1):objwind(1)+objwind(3),[1 2 3]);
tsl=0.05;
tsh=2;
tbl=0.05;
tbh=0.95;
bins=32;
boxresize=1.6;
component=1;
h=BuildHistogram(reg,tsl,tsh,tbl,tbh,1, bins);
h=h/max(h(:));
abstime=tic;

%%%%
w=rec2w(objwind);
ratio= (w(2)-w(1))/(w(4)-w(3));
map=ProbabilityMap(frame, h, tsl, tsh, tbl, tbh, component, bins);
w=rec2w(objwind);
 w = CamShift(map, w, 10,1,boxresize, ratio);
objwind=w2rec(w);
xc = round(w(1) + (w(2)-w(1))/2);
yc = round(w(3) + (w(4)-w(3))/2);
confidence=mean(mean(map(objwind(2):min(120,objwind(2)+objwind(4)),
objwind(1):min(160,objwind(1)+objwind(3))));
% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
  % hObject    handle to checkbox1 (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    structure with handles and user data (see GUIDATA)
  % --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
  global t0;
  global timesave;
  global xcsave;
  global ycsave;
  exname='exp_';
  exname=[exname, date];
  c=clock;
  exname=[exname,'_',num2str(c(4)'),'h',num2str(c(5)),'m.mat'];
  save(exname,'t0','timesave','xcsave','ycsave');
  % hObject    handle to pushbutton13 (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    structure with handles and user data (see GUIDATA)
  % --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
  % hObject    handle to pushbutton14 (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    structure with handles and user data (see GUIDATA)
  system('pario 12');
  % --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
  % hObject    handle to pushbutton15 (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    structure with handles and user data (see GUIDATA)
  global h;
  uisave('h');
  % --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
  % hObject    handle to pushbutton16 (see GCBO)
  % eventdata  reserved - to be defined in a future version of MATLAB
  % handles    structure with handles and user data (see GUIDATA)
global rect_selected;
global objwind;
global data;
global h;
global abstime;
rect_selected=0;

[fil, path, filtind] = uigetfile;
if filtind==1
    load([path,fil]);
    objwind=[5 5 155 115];
    abstime=tic;
    rect_selected=1;
else
    rect_selected=0;
end
Bibliography


