DESIGN AND DEVELOPMENT OF A TARGETING SYSTEM USING MATLAB

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

_____

Stephen Kyle Murphree

Certificate of Approval:

_____	_____
Jeffrey W. Fergus	Ruel A. Overfelt, Chair
Associate Professor	Professor
Materials Engineering	Materials Engineering
_____	_____
Barton C. Prorok	George T. Flowers
Associate Professor	Dean
Materials Engineering	Graduate School

DESIGN AND DEVELOPMENT OF A TARGETING SYSTEM USING MATLAB

Stephen Kyle Murphree

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Degree of

Master of Science

Auburn, Alabama
August 9, 2008

DESIGN AND DEVELOPMENT OF A TARGETING SYSTEM USING MATLAB

Stephen Kyle Murphree

_____
Signature of Author

_____
Date of Graduation

VITA

Kyle Murphree was born on March 26, 1982 in Homewood, Alabama. He is the son of Steve and Marilyn Murphree. In 2000, he graduated 2nd in his class of 100 from Guntersville High School in Guntersville, AL with an Advanced Academic Diploma. After high school, he enrolled at Vanderbilt University. In 2005, he graduated from Vanderbilt with a Bachelor of Engineering degree in Chemical Engineering. In 2006, he entered Auburn University to pursue a Master of Science degree in Materials Engineering.

THESIS ABSTRACT

DESIGN AND DEVELOPMENT OF A TARGETING SYSTEM USING MATLAB

Stephen Kyle Murphree

Master of Science, August 9, 2008
(B.E., Vanderbilt University, 2005)

132 Typed Pages

Directed by Ruel A. Overfelt

Advances in security equipment and technology have always been essential to prevent malefactors from performing criminal acts. Therefore, it is essential to develop new ways to deter wrongdoers. It is the intent of this project to design a targeting system that will be the precursor of a security system that will be able to target a criminal and deliver a payload to thwart their activity. The system, whose main components are a USB camera and a RS-232 Pan/Tilt Device, is controlled using a program written in Matlab.

Initially, the geometry of the system's environment had to be modeled as the basis of the control program. Once the geometry had been modeled, two programs were written to control the system. The first program targeted stationary objects fixed to a wall. The second program tracked a moving LED light.

In addition to writing a program to control the system, the camera had to be calibrated to correct for distortion in the image. A calibration technique was developed

to account for any distortion in the lens. This calibration was performed at a short distance to test the technique. This technique was then applied to the actual system environment that had a much greater distance from the target and the camera.

After the calibration had been performed, tests were run to test the systems ability. The first test targeted a matrix of pushpin pins fixed to a wall. The second test tracked a LED light fixed to a moving pendulum. In addition, the system's performance, i.e. the time it took to perform the computing necessary to operate the hardware, was tested for both cases.

ACKNOWLEDGMENTS

First, I would like to give thanks to Almighty God for his blessings, mercy, and grace as well as His Son Jesus, through whom salvation is possible. I would also like to thank Dr. Tony Overfelt for his advice and guidance throughout my graduate career. I would also like to thank Dr. Jeff Fergus and Dr. Bart Prorok for serving on my committee and for their advice and guidance.

I would like to thank Ben Beyer, Taylor Owens, and Joseph Cali for their invaluable assistance and advice with this project. I would like to thank my family, whom I love very much, for their moral and financial support throughout all of my academic endeavors.

I would especially like to thank my beautiful and wonderful wife, Laura, not only for her assistance in this project but also for her selfless sacrifice, encouragement, and love throughout my graduate career. I would not have been able to make it through this process without her by my side. I love you sweetheart.

Style manual or journal used: Auburn University Guide To Preparation And Submission Of Thesis And Dissertations. Bibliography is IEEE format.

Computer software used: Microsoft Word 2003, Microsoft Excel 2003, Matlab 7.3.0

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

These days it is almost impossible to turn on a news broadcast or open a newspaper and not see a story about a person who was murdered, robbed, or raped. These senseless acts of violence are, unfortunately, commonplace in our culture. Even though humanity has made advancement towards a more civilized society, mankind will never be rid of the darker element of human nature. In 2007, almost 2 million violent crimes were committed in the US alone [1, 2]. As a result, there is a constant need to make advancements in the areas of security and protection of the innocent.

The September 11 terrorist attacks were one of the worst acts of violence in this country's history. However, imagine for a moment if there had been a way to prevent the attacks from happening. As the terrorist attempts to enter the cockpit, the pilot secretly aims a device at the intruder. The pilot fires a taser at the terrorist, sending several thousand volts of electricity through his body effectively neutralizing him. Or imagine a hoodlum with a gun trying to rob a convenience store clerk in the middle of the night. An operator monitors the situation on a video screen in a separate location, perhaps miles away, aims a device at the assailant and fires a beanbag at him. This blow breaks his arm causing him to drop his gun and flee the scene before he can cause any physical harm to the clerk.

The development of a system similar to the ones described above would be a tremendous asset to those looking to protect themselves from people whom mean to cause harm. It is the goal of this project to develop an inexpensive and effective precursor to such a system. In order to accomplish this, the project detailed by this thesis has two main objectives. The first objective is to develop a system with the capability to target a stationary object by aiming a laser pointer as accurately as possible. The second objective is for the system to be able to track a simply moving object with the same laser pointer.

2. LITERATURE REVIEW

2.1 Security Applications of the Main System Components

The component of the proposed system that is most commonly used in security applications is the camera. Security cameras are widespread and used to monitor places such as private homes, financial institutions, government buildings, schools, and border crossings. Security cameras can be acquired through several commercial outlets and are available from very basic models to cameras with enhanced features such infrared capability.

Security cameras can be coupled with video recording equipment so that the incoming video image can be reviewed later. This recording can then be used as evidence against someone who performs a malicious or criminal act such as trespassing or robbing a bank.

Pan and tilt devices can also be used in tandem with cameras to move them in order to monitor a larger area. For example, Science International Applications Corporation developed a monitoring system for detecting intruders along the perimeter of secured area [3]. It incorporates a series of cameras, a pan and tilt device, motion detectors, a computer aided tracking system, and a spot light [3]. When an intruder is detected, the system will acquire the intruder and track their movements [3]. The

spotlight, mounted on the pan and tilt device along with the camera, allows the security personnel to see the intruder in the dark.

2.2 Image Distortion

2.2.1 Overview of Optical Aberrations

In order to create a system that can take image data and accurately produce a desired physical response, all aberrations in the images must be taken into consideration. Optical aberrations are defects in an image produced by a lens, e.g. an image produced by a lens may contain a region that is out of focus. These defects occur in all lenses [4]. There are seven main types of aberrations: spherical aberration, coma, field curvature, astigmatism, distortion, axial color, and lateral color [4]. The first five are monochromatic aberrations and the last two are chromatic aberrations. While it is not necessary to discus all of these in detail, spherical aberration will be discussed as an example. Spherical aberration arises from a deviation in paraxial approximations used for lens design. [5] This results in not all incoming light rays focusing (mainly those at a greater distance from the optical axis) at the focal point which causes parts of the image to be blurry and out of focus [4]. This is shown in Figure 2.1

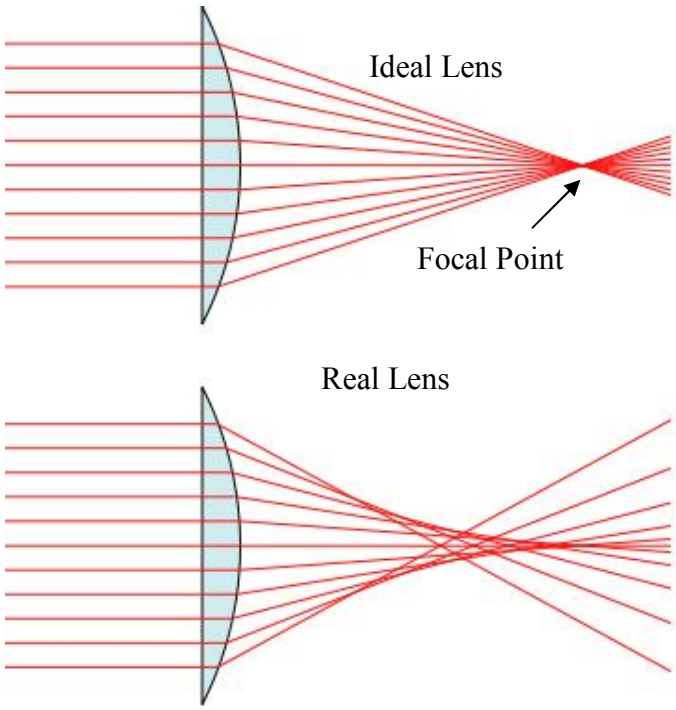

Figure 2.1: Light rays being focused in an ideal and real lens which shows spherical aberration [6]

While it is possible to create a lens that produces a perfect image from a single image plane, it is not possible to create a lens that produces perfect images from multiple image planes [7]. This is of particular interest to camera lenses that focus objects on several image planes. Cameras account for spherical aberration by using of an adjustable aperture [5]. A reduction in the aperture size only allows light to pass through the center portion of the lens; this effectively blocks the light rays that are not passing through the center of the lens [5]. This produces sharper images.

2.2.2 Distortion

General-use cameras, such as the one used in this project, are designed to focus over a wide range; therefore, aberrations that cause blurring, such as spherical aberration, are corrected and shall be considered negligible. Also, chromatic aberrations will have no negative effect on this project. However, distortion is an unique phenomena in that it does not result in an image that is blurred; instead, distortion results in the image becoming stretched or compressed, i.e. the image is scaled inhomogeneously [4, 8]. This is the result of the focal length of a lens varying "as a function of field angle or image size" [4]. There are two common types of image focus distortion: barrel and pincushion [4, 8]. Examples of these are shown in Figure 2.2.

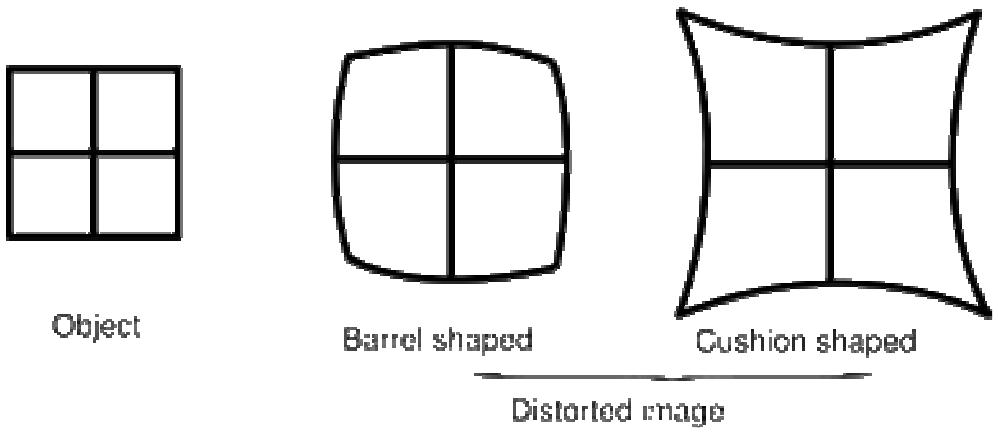

Figure 2.2: The two common types of image distortion [9]

Since the scaling of the image is inhomogeneous, any distortion in the image will have to be characterized and corrected if accurate physical distance measurements are going to be deduced from an image.

2.2.3 Distortion Correction

Since image distortion is to be expected, a method to correct for this aberration needs to be implemented. However, research on calibration techniques include very complex mathematics and are developed for precise three dimensional computer vision systems [10, 11]. This is more complicated than the current system warrants; therefore, a simplified calibration technique was developed in this project.

2.3 Computer Vision

2.3.1 Introduction

Computer vision, also known as machine vision, image understanding, and robot vision, is an area of study that involves developing machines with vision systems that allow them to accomplish various tasks [12]. The concept for this as been around since the 1930's when Electronic Sorting Machines company developed automated food sorting machines [13]. This technology has evolved into a complex field that implements computer systems and robotics to perform tasks from welding to detecting defects in printing operations to determining the correct portions of packaged foods [13].

2.3.2 Geometric Computations

Textbooks written on geometrical computations necessary for most aspects of computer vision contain very complex geometric computations in the areas of projective

geometry, matrix translation and rotation, and analytical conics as well as statistical analysis of these computations [12]. These geometric computations are used to gather information about a three dimensional environment from a two dimensional image. It is beyond the scope of this thesis to delve into this in detail. However, future generations of this system will require much more complicated mathematics than those used in this project since this project was limited for the sake of simplicity.

2.3.3 Image Analysis Research in Computer Vision

Thousands of articles and books have been written to describe research done in computer vision. It is a versatile field with numerous applications. However, this section will focus on a few research papers that describe research in image analysis that has the potential to be useful to future versions of the system developed for this thesis.

One area of research is determining shape and color of agricultural products to aide in sorting. For example, a study funded by the Malaysian government developed a technique to determine the quality of starfruit [14]. While their shape analysis may not be useful for this project, their color analysis algorithm was 92% accurate, which was found to be better than human inspection [14]. This type of computer vision analysis has also been utilized in the paper industry to locate defects [15].

This kind of analysis could be accomplished by writing similar algorithms to detect colors (say of a shirt) and track that color as it moves in the image space by using an additional algorithm. In the interest of simplicity, a commercially available camera could be purchased that has color tracking capabilities. However, many papers have been

written on computer vision color analysis. Therefore, the option is available for a custom color tracking system to be developed.

Researchers at the Southwest Research Institute have coupled security cameras with computer vision to aid security officers in locating suspicious activity [16]. Previous detection systems were very simplistic and were easily set off by false triggers [16]. Recent algorithms that have been written for the system eliminate false triggers, such as moving foliage, and alert security to movement by a person or a vehicle [16].

2.4 Spatial Tracking Systems

An image system that could distinguish a human body from the environment, determine its position relative to the camera, i.e. facing the camera, turned sideways, bending over, etc., and offer specific, predetermined aiming points for that position would give a tremendous advantage to the operator. Situations that would call for a system like this demand quick, decisive action; therefore, any automated assistance that the system could give could mean the difference between life and death. Therefore, it is advantageous to study the literature for current techniques used in spatial tracking to gauge the feasibility of this system's development.

One popular technique used today is motion capture. This process involves using a series of markers, either optical or magnetic, attached to a test subject [17]. A system of cameras is then used to track the motion of the markers as they move inside a volume defined by the system. This technology is used in both animation and biomechanics [17]. Examples of biomechanical studies are studying the kinematics of a human arm reaching

for an object [18] and determining the preferred transition speed from a person walking to a person running based on forces on the ankle [19]. As innovative as this technique is, it is obviously impractical for this application.

Fortunately, there has been research done in marker-less motion capture to model a human being in a three dimensional space. One technique estimates anthropometry and pose from a single image by comparing the human to a landmark in the image [19]. The image coordinates of the landmark are previously determined before real time measurements commence [19]. Anthropometric statistics are then used to help the computer model estimate body size [19]. Another approach uses predefined templates to construct a model of the subject [20]. This approach was shown to be not only practical, but it also created better models than the marker systems due to the fact that two orders of magnitude more data were able to be collected [20]. While this type of imaging is well beyond the scope of this project, it is clear that the technology exists to aid in the completion of future versions of this system.

2.5 Non-lethal Weapons

2.5.1 Introduction

In 1985, the Supreme Court of the United States of America ruled in Tennessee v. Gardner that lethal force was to be limited when pursuing a fleeing suspect [21]. The court stated that lethal force could only be used when the officer believed the suspect to have intentions to use lethal force on an officer or bystander [21]. As a consequence of this, viable alternatives had to be developed. Beginning in the 1990's, non-lethal

weapons became an area of serious investment and research by both the military and civilian police organizations [22]. As advances in this area are made, security systems, such as the one envisioned in this project, stand to benefit greatly as they will offer more versatility.

2.5.2 Traditional Non-lethal Weapons

Non-lethal weapons have been used by police and the military for decades as a means of crowd control and suspect apprehension. Non-lethal weapons first appeared in the 1910's in the form of chemical irritants which were used by policemen in France [21]. By the 1920's, the United States had begun manufacturing them for their own police forces [21]. During the 1930's, electroshock weapons were being used in Argentina, although they were mainly used for torture and interrogation [21]. Police forces in the United States were using electroshock weapons in the 1950's as a means of riot control [21]. Another traditional non-lethal weapon used by police are impact projectiles. These were developed in Hong Kong as wooden bullets fired at the legs of suspects since firing them at the torso or head would have more than likely led to death [21].

These weapons have evolved significantly over the decades. Chemical irritants have gone through several formula changes ultimately becoming the capsicum based weapon, or pepper spray, that is used today. Electroshock weapons have grown from rudimentary cattle prods to complex stun guns and tasers, which are projectile electroshock weapons. Wooden bullets have been replaced by rubber projectiles and beanbags. Examples of these rounds are shown in Figure 2.3.

Figure 2.3: A rubber projectile (on the left) and a beanbag projectile (on the right) [22]

These non-lethal weapons could all be employed in future versions of the proposed system. Chemical irritants could be delivered through commercially available paintball systems. Rubber projectiles, beanbags, or electroshock projectiles could also be deployed remotely using existing technology.

2.5.3 Current Innovations in Non-Lethal Weapons

As more research and development of non-lethal weapons is performed, advances in technology will lead to systems that were unimaginable in the past. One such system is the army's Active Denial System (ADS). This system sends electromagnetic waves at

an enemy causing the water molecules to vibrate and heat up [23]. This results in a burning sensation which causes a person to get out of the way of the beam but causes no permanent damage [23]. Currently, this system is very large and is mounted on an armored vehicle. While it may be several decades before this technology is miniaturized enough to fit on a practical security system, advances in technology may one day make that a reality.

Another advance in non-lethal weaponry involves light-emitting diode incapacitators. This devices cause the person looking at the beam to become disoriented, nauseated, or temporarily blind [24, 25]. Again, this weapon does no permanent damage. Currently, prototypes of this weapon are flashlight-type devices as shown below in Figure 2.4

Figure 2.4: Prototype light-emitting diode incapacitator [25]

This type of weapon would be ideal in a close-quarters situation where the perpetrator would be looking in only one direction. This technology only illustrates the advances that non-lethal weapons have made in causing as little physical damage to a suspect as possible.

Long range acoustic devices are another technology being pursued. These devices emit sound waves at 150 decibels [26]. The waves travel in a narrow beam so those not directly in the beams path will not be affected by the sound waves [26]. These waves are emitted in short bursts so that no permanent hearing loss is experienced.

3. EXPERIMENTAL SYSTEM DEVELOPMENT

3.1 System Overview

3.1.1 Hardware

The hardware setup for this project is relatively simple and inexpensive. The components that make up the entire system are a pan and tilt device with a digital controller, a 3A DC power source, a camera, and a personal computer.

The pan/tilt device is a Directed Perception PTU-D46-17. Two stepper motors control the motion: one pans horizontally and one tilts vertically. Panning and tilting is controlled by a Directed Perception model PTU-D46E controller through a 15 pin serial cable. The controller is connected to the host computer using a 9-pin RS-232 cable. The power supply (Radio Shack 22-507) is connected to the controller through a 2.1/5.5mm coaxial cable. The controller houses the interfaces for RS-232 and RS-485, relays power from the power supply to the pan/tilt device, and is capable of being networked to other controllers in the case of a multiple pan/tilt device system. The controller also has multiple serial ports to connect a joystick or trackball for manual control of the pan/tilt device. Figure 3.1 shows a picture of the power supply and the digital controller.

Figure 3.1: Power supply and controller

The pan/tilt device is able to carry up to a 6 pound payload attached to a platform on top of the device. For this project, the payload is an ordinary laser pointer. The pan/tilt device responds to either ASCII or binary commands sent from the host computer and moves in increments of .0514 degrees in both the pan and tilt directions. The technical specifications for this device are given in the Appendix. A picture of the pan/tilt device is shown below (Figure 3.2).

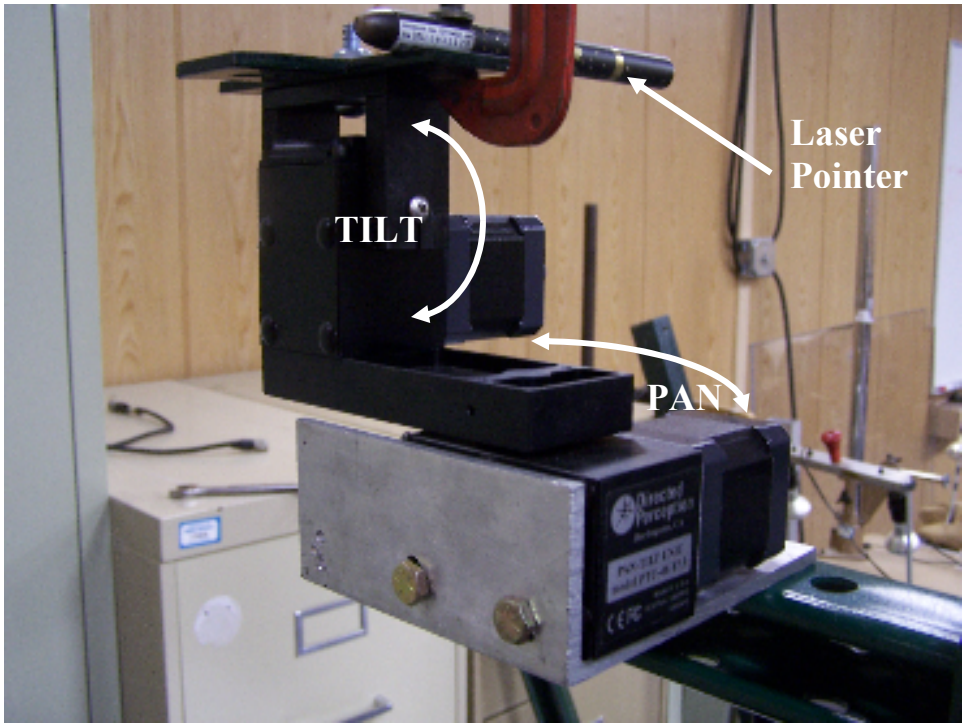

Figure 3.2: Pan/tilt device with a secured laser pointer

The camera used in this project is a Phillips SPC 700NC PC camera which is a 1 megapixel, cmos image sensor camera that uses a standard and easily interfaced USB connection. This camera can process images at frame rates of 5, 10, 15, 20, 25, and 30 frames per second. Figure 3.3 contains a picture of the camera, Figure 3.4 shows a schematic of the entire system, and Figure 3.5 shows a picture of the pan/tilt device in relationship to the camera.

Figure 3.3: Phillips SPC 700NC PC camera

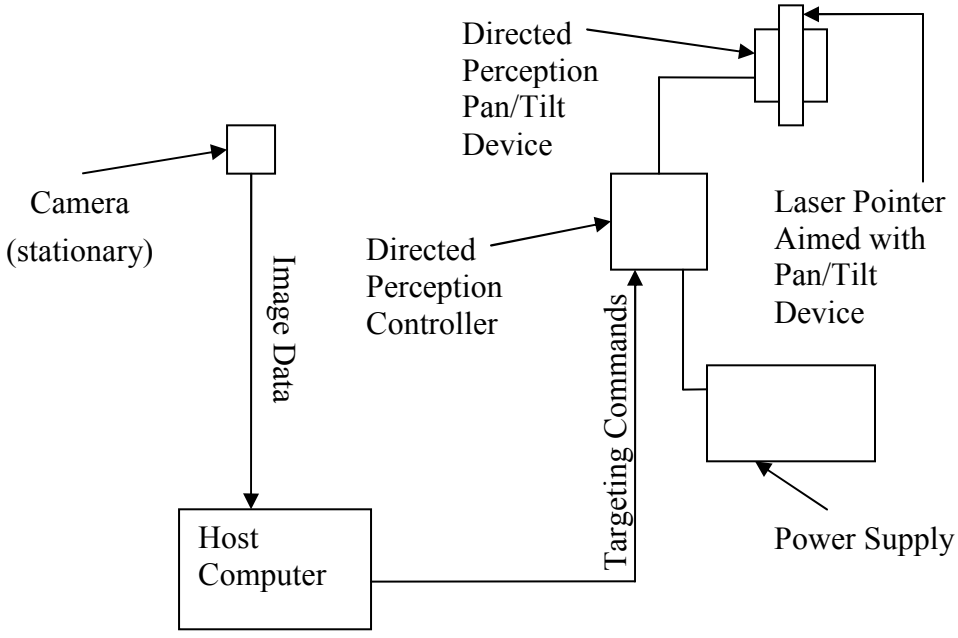

Figure 3.4: A schematic diagram of the system

Figure 3.5: Pan/tilt system in relationship to the camera

3.1.2 Software

3.1.2.1 Matlab

Matlab 7.3.0 was selected for the overall control software development due to its versatility. Matlab is an excellent tool for prototyping because of the ease in which programs can be written; also, the number of functions Matlab contains can be used to perform a plethora of tasks. The programs that Matlab runs (both those factory installed and user generated programs) are called "m-files". Matlab also has several add-ons available called toolboxes. The toolboxes allow Matlab to perform tasks for several applications such as signal processing, symbolic math, data acquisition, etc. The specific toolboxes used in this project are the Image Acquisition, Image Processing, and Curve Fitting Toolboxes.

The pan/tilt system and camera must be integrated together by bringing both devices into the Matlab environment. The code to accomplish this is given in the Appendix.

3.2 Geometric Modeling

3.2.1 Introduction

In order to aim the pan/tilt device, the geometry of the environment in which the system is located must be modeled. This model was used as the basis for the code that determines the direction and angle that the pan/tilt device must move in order to be properly aimed based upon the image data from the camera.

3.2.2 Geometric Modeling

3.2.2.1 Pan and Tilt Movement

Due to the nature of panning and tilting, the environment surrounding the pan and tilt device is most easily modeled using a spherical coordinate system. Points represented by the more familiar Cartesian coordinate system for any spatial environment can be converted into spherical coordinates using the following equations:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{3-1}$$

$$\theta = \arctan\left(\frac{\sqrt{x^2 + y^2}}{z}\right) \tag{3-2}$$

$$\varphi = \arctan\left(\frac{y}{x}\right) \tag{3-3}$$

where r is the radial distance to a point measured from the origin, θ is the zenith angle, φ is the azimuth angle, and x, y, and z are the coordinates of the same point in the Cartesian Coordinate system. However, since the present system uses only one camera, distance from the camera to the target image plane must be known. This effectively limits the location of the target to a 2 dimensional plane. This limitation was purposefully imposed on the system for the sake of simplicity. Clearly, a multi-camera system would be required in order to stereographically determine distances.

Because of this limitation, the movement necessary to aim the pan/tilt device to target an object on an image plane can be determined very easily. Movement in both the horizontal (pan) and vertical (tilt) directions on the plane can be modeled using the same trigonometric relationship. These models used for movement of both components are shown in the following figures (Figures 3.6a and 3.6b).

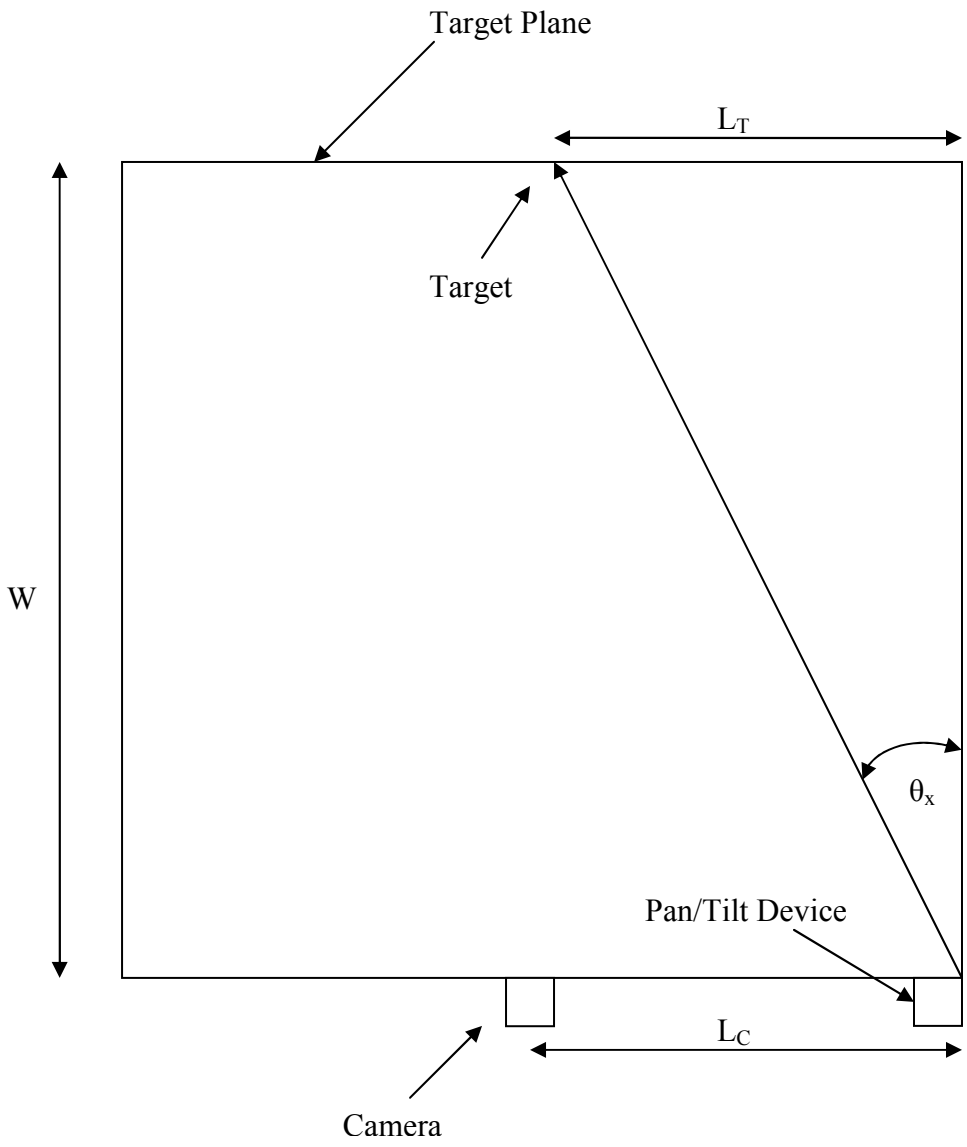

Figure 3.6a: Geometrical model describing the motion of the pan portion of the pan/tilt device (view from above)

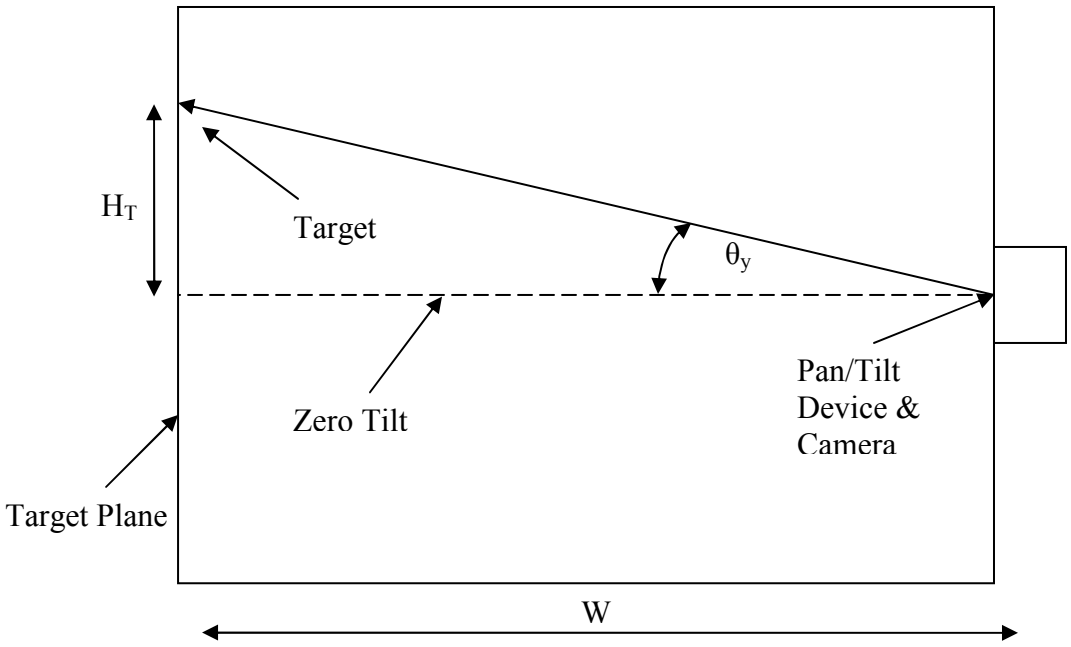

Figure 3.6b: Geometrical model describing the motion of the tilt portion of the pan/tilt device (horizontal view)

The number of degrees of pan of the pan /tilt device (θ_x) to aim at the desired location is determined by the following equation:

$$\theta_x = \tan^{-1}\left(\frac{L_T}{W}\right) \tag{3-4}$$

where L_T is the horizontal distance to the target and W is the perpendicular distance from the system. Likewise, the number of degrees of tilt of the pan/tilt device (θ_y) is determined by the following:

$$\theta_y = \tan^{-1}\left(\frac{H_T}{W}\right) \tag{3-5}$$

where H_T is the vertical distance to the target from the horizontal plane containing the camera and the pan/tilt system.

3.2.2.2 Hyperbolic Corrections for Tilt Movement

Unlike the pan portion, tilt movement needs additional considerations to accurately model its motion. This is due to the pan/tilt device creating an imaginary cone as it rotates horizontally while the device is tilted up or down at an angle. If any desired image plane intersects the cone determined by a panning tilted laser pointer, a hyperbola will be generated as the laser pans across the image plane. Therefore, this conic section must be taken into account. To do this, a program was written to model the hyperbolas created by a plane intersecting a double cone with an arbitrary angle. Examples of the cone and the hyperbolas created are given below in Figures 3.7a and 3.7b.

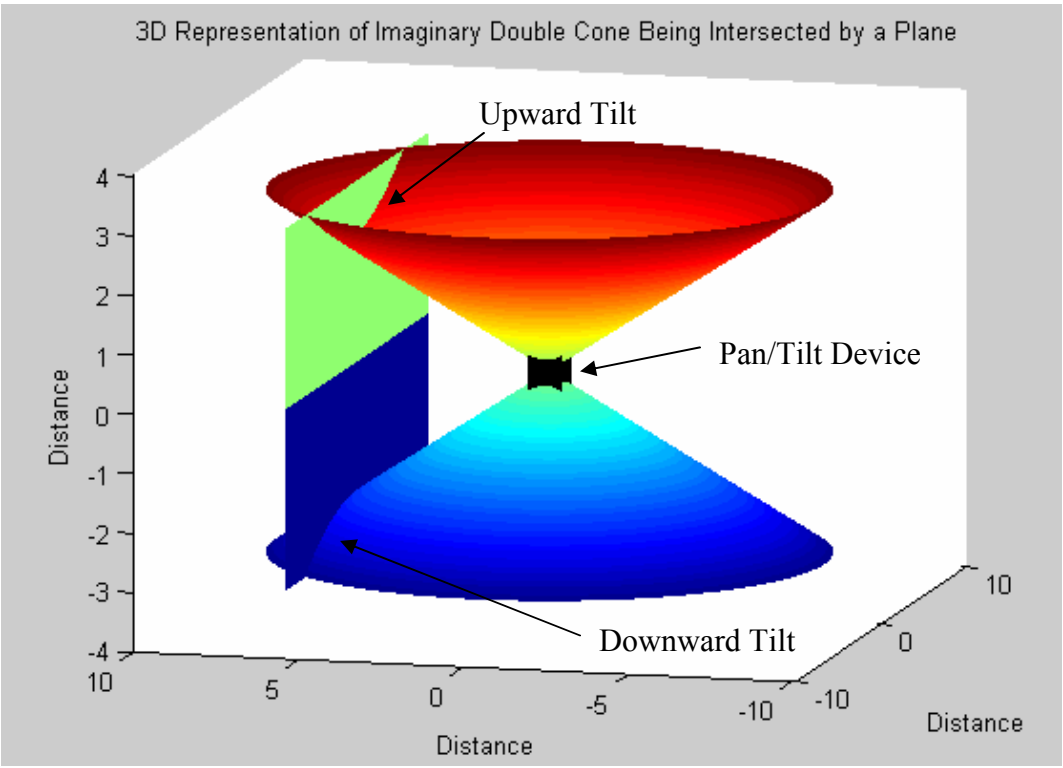

Figure 3.7a: Imaginary double cone, created from upward and downward tilt of pan/tilt device, being intersected by a plane

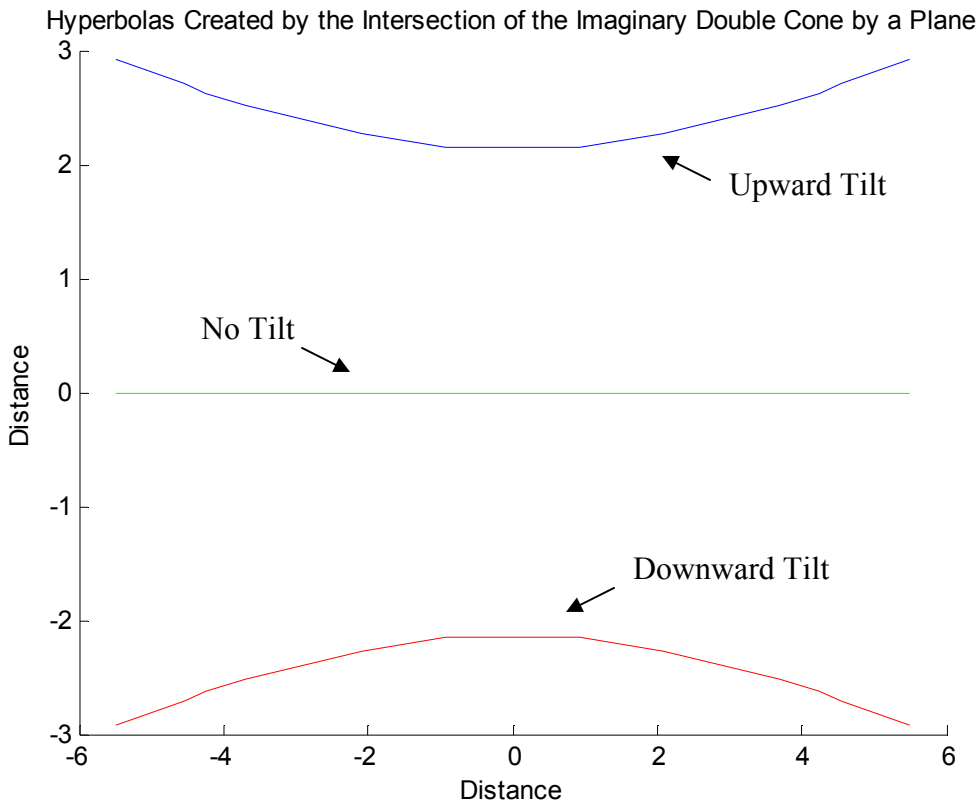

Figure 3.7b: Hyperbolas created from intersection of double cone by plane

From the two hyperbolas created, the one that pertains to the current position (depending on whether the pan/tilt device is tilted up or down) is then treated as a simple parabola. A correction can then be made to the number of degrees needed by either adding (when the pan/tilt device is tilted downward) or subtracting (when the pan/tilt device is tilted upward) the difference from the two models at any horizontal distance from the pan/tilt device.

3.3 Software Development

3.3.1 Introduction

Once the system's environment has been modeled, the code controlling the movement of the system can be created. This code had two goals: bring an image into the Matlab environment and determine the motion necessary to aim the laser pointer at a desired location. For this project, two programs were written to accomplish these goals. The first program targets a static object inside the system's environment by moving the pan/tilt device to place a laser on the targeted object. The second program, which also uses a laser, tracks an object as it moves within the environment. The goal of the second program is the same as the first, i.e. to place the laser on the targeted object.

In order to aim the laser pointer correctly, the programs also had to make two corrections. The first correction was the hyperbolic correction mentioned above. Another correction was needed to correct for optical distortion. This will be discussed in Section 3.4.

3.3.2 Static Targeting

The static targeting program is designed to give an operator the ability to target an object within the field of view of a camera. As mentioned earlier, because of the use of a single camera, this program requires that the perpendicular distance to the target be known. By taking an input from the operator, the number of degrees needed to move the

pan/tilt device to the target can be calculated for the other two dimensions using the geometrical models previously discussed.

Before the program can be run, several parameters must be set. The first parameter is the perpendicular distance to the target. The next two parameters that must be set are the vertical and horizontal components of the image pixel that represent the location of the light from the laser pointer when the pan/tilt device is at zero degrees pan and zero degrees tilt. This was determined using the image tool in Matlab.

The next required parameter is the physical distance to pixel ratio. This ratio is defined by a unit of length divided by the number of pixels that represents that distance in the video object. For example, if 5 meters in real space is displayed as 400 pixels, then the ratio would be 0.0125 meters/pixel.

Once these parameters are determined, the program is ready for operation. Running the program creates a video object and displays it on the monitor. The previously mentioned parameters are called into the environment and a serial object is created. The program is then ready for an input from the operator. The program pauses its operation using the *waitforbuttonpress* function. The operator then identifies the target using the cursor and clicks on its image in the figure created by the program. By clicking the mouse inside the video object, Matlab extracts the location of the pixel of the mouse click. That pixel information is the input for the calculations used to determine the distance of this pixel from the pixel of zero degrees of pan and tilt. This pixel distance is then converted to a physical unit of measure by multiplying it by the distance to pixel ratio. By doing this for both the horizontal and vertical component of the pixel input, the degrees to the target spot can be calculated. From these camera angles, the

degrees needed to then move the laser pointer to the desired location are determined. These degrees are then converted into position commands for the pan/tilt device.

The program then proceeds to correct for the hyperbola created from the geometry of the system. After the program calculates the appropriate hyperbolic correction, an equation is then fit to the hyperbola. This equation is then used to determine a tilt angle which is converted into a position command. This position command is then combined with the position command determined from the first model.

After the position commands are determined, a calibration matrix is called. This matrix serves to correct for errors associated with image distortion. The correction is added or subtracted to the previously determined position commands. These corrected commands are then converted into an ASCII format and sent to the pan/tilt device. The device then moves accordingly, and the system awaits another input. To help illustrate the process, a flowchart of the program is provided in the following figure (Figure 3.8). The code for this program can be found in the Appendix.

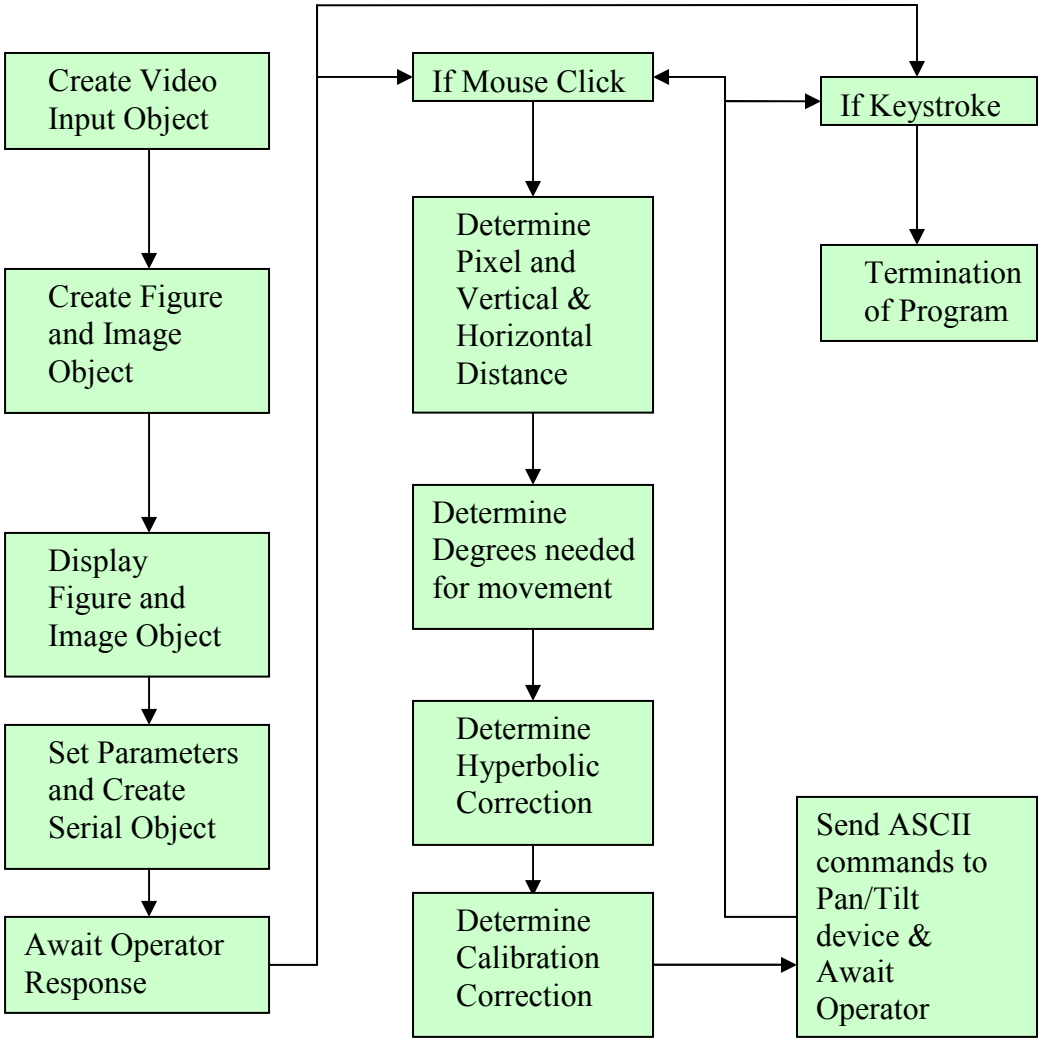

Figure 3.8: Flowchart describing the static targeting program

3.3.3 Dynamic targeting

The dynamic targeting program is very similar to the static targeting program except for a few key differences. First, an actual snapshot from the video object has to be imported into the software loop that controls the pan/tilt device. Second, the input into the program is determined from the imported snapshot image. Specifically, the input is the centroid of an object that the operator wishes to track where the centroid is the "center of mass" of the pixels that represent the object. From this input, the position commands needed to successfully aim at the target are calculated.

Since the process is automated, the program determines the next input as soon as it has sent the ASCII command to the pan/tilt device. This continues continuously until the program has processed the number of frames set by the operator. Another flowchart is provided to illustrate this process (Figure 3.9). The code for this program can be found in the Appendix.

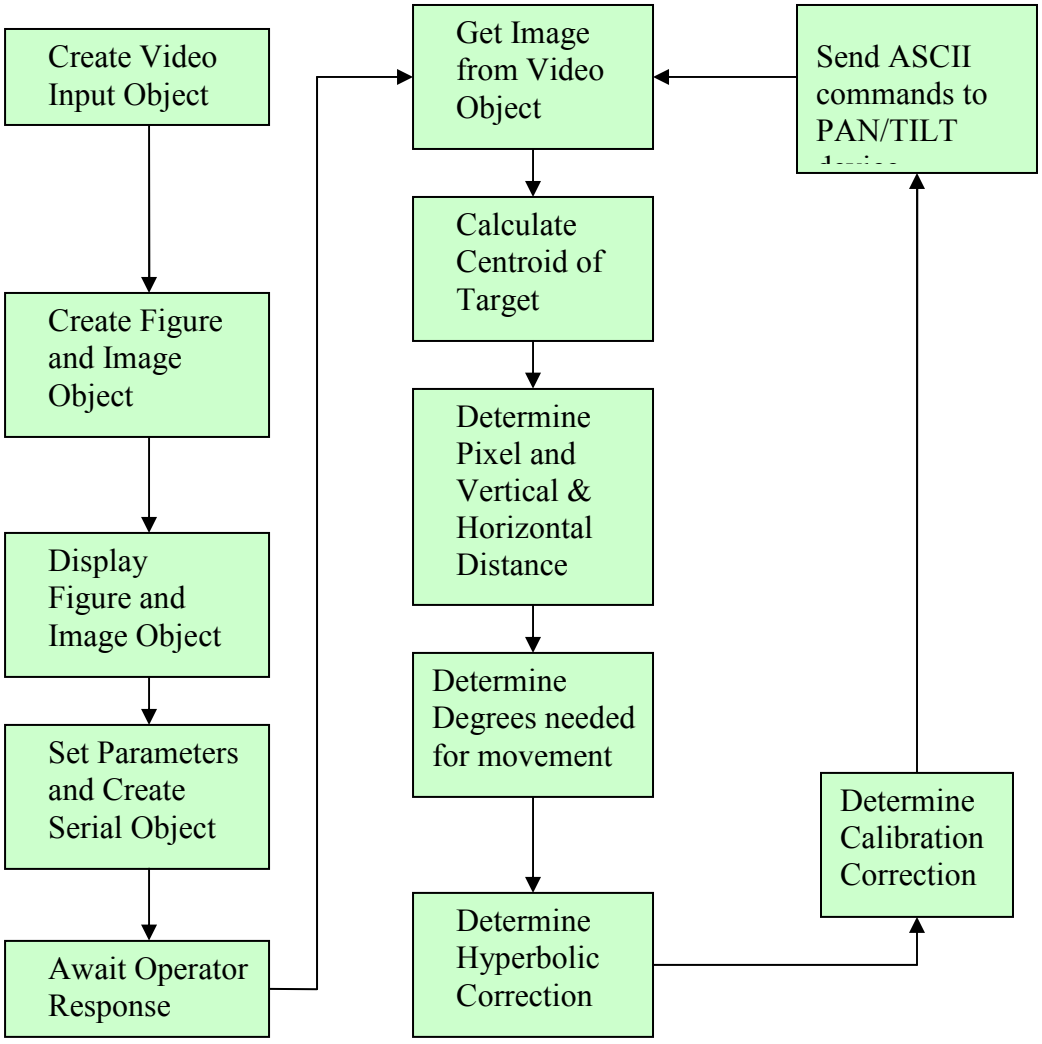

Figure 3.9: Flowchart describing the Dynamic targeting Program

3.4 Camera Calibration

3.4.1 Purpose and Objective

As stated earlier, the lens must be tested to determine the amount of optical distortion. In order to accomplish this, the camera must first be properly aligned to make sure that the photo detector in the camera is perpendicular with the target. It was not

assumed that the image sensor was parallel with the housing of the camera. To accomplish proper alignment, a target image was acquired as the camera was panned and tilted. The pixel length of the target image was measured at these different locations.

As the target image deviates from perpendicular alignment with the image sensor, the number of pixels representing it will increase. The pixel length will decrease as the target image approaches alignment and then increases again as it moves away from alignment. Proper alignment is achieved when both the horizontal and vertical pixel lengths are minimized.

Once the camera is aligned, the amount of distortion can be determined by measuring a matrix of identical targets that fill the acquired image. A calibration matrix can then be created and applied to an image to correct for the optical imperfections of the camera. For this experiment, calibration was first performed at a short distance to develop the calibration technique. The calibration was then repeated at a longer distance when the system was set up for testing targeting capability.

3.4.2 Short Range Calibration

3.4.2.1 Experimental Setup

The camera was first secured to the top platform of the pan/tilt device using a ¼-20 bolt. The pan/tilt device was bolted onto an aluminum brace which was secured onto a Unistrut frame. This frame was fastened to a steel table that was .914 meters per side. The system was then attached to the computer as described in Chapter 3. Directly across from the pan/tilt device on the other side of the table was a poster board that contained

the image to be measured. A picture of this arrangment is given below in Figures 3.10 and 3.11.

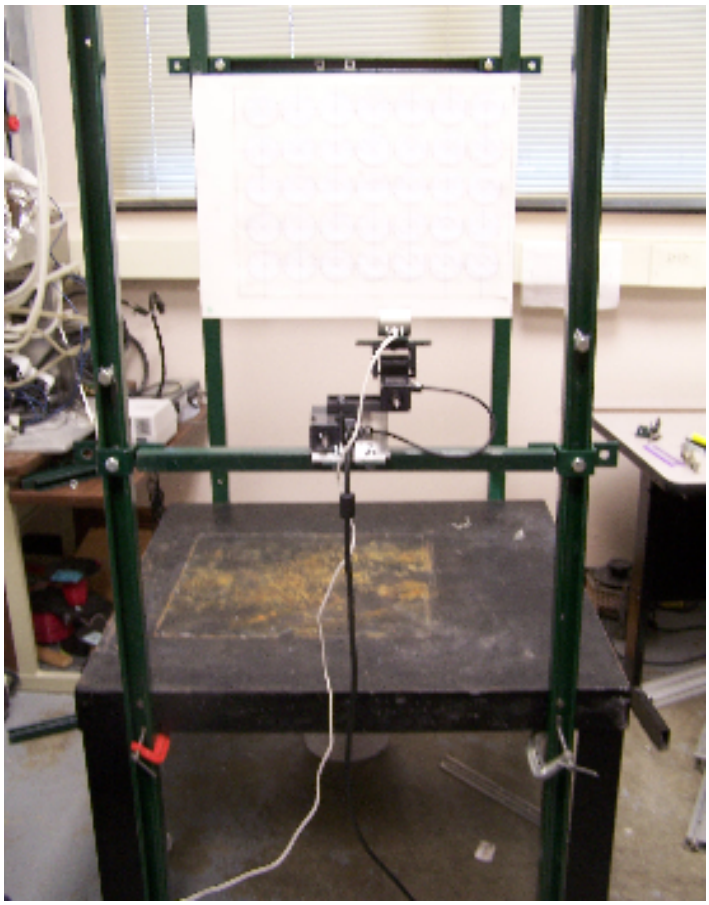

Figure 3.10: Experimental arrangement for short range calibration. Note that the camera is mounted on the pan/tilt device.

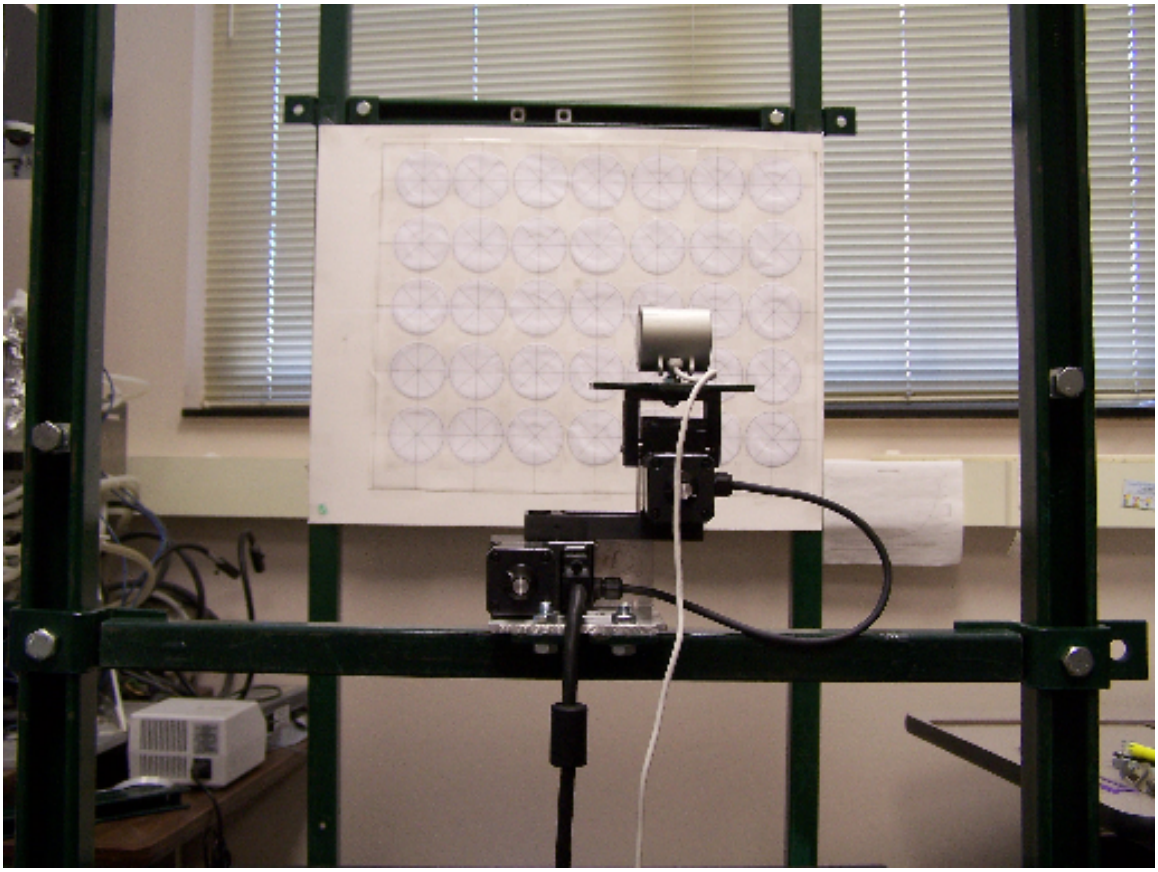

Figure 3.11: Close-up of the experimental arrangement for short range calibration showing the calibration image

3.4.2.2 Experimental Procedure

As previously stated, in order to determine alignment between the target and the image detector, measurements of the target image were taken at different degrees of pan and tilt. At each new position, an image of the poster board was acquired and measured using the Matlab software distance tool. This tool measures the number of pixels between two locations in the image. To determine camera alignment, the target image was a circle whose initial position was the exact center of the image. The circle was

measured across its diameter vertically and horizontally; an average of these two diameters was then taken. The following figure shows a picture of the target image and a measurement across the horizontal diameter (Figure 3.12). Only the center circle was used to align the camera.

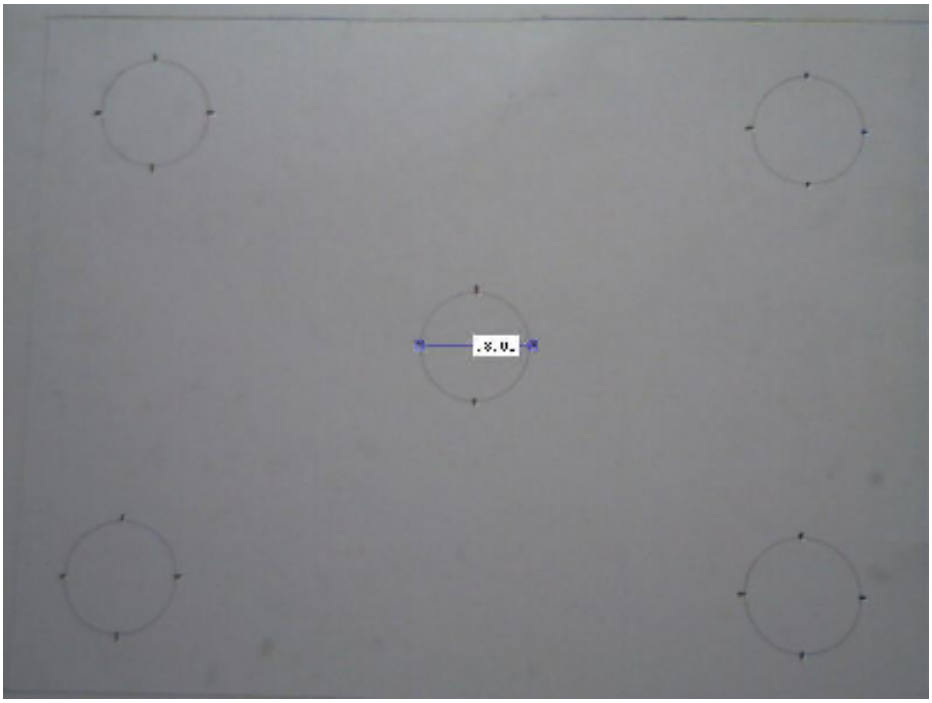

Figure 3.12: Target image with the horizontal diameter being measured by the distance tool

After the camera was properly aligned, a new target was placed on the Unistrut structure. This target image consisted of a matrix of 35 identical circles. These circles were then measured across four diameters. After averaging these diameters, the average diameter of the each circle was then used to create a 7 by 5 matrix of image correction factors. These factors were used to correct the distortion for the entire field of view of

the image. The following figure shows a picture of the matrix of circles used for this part of the experiment (Figure 3.13).

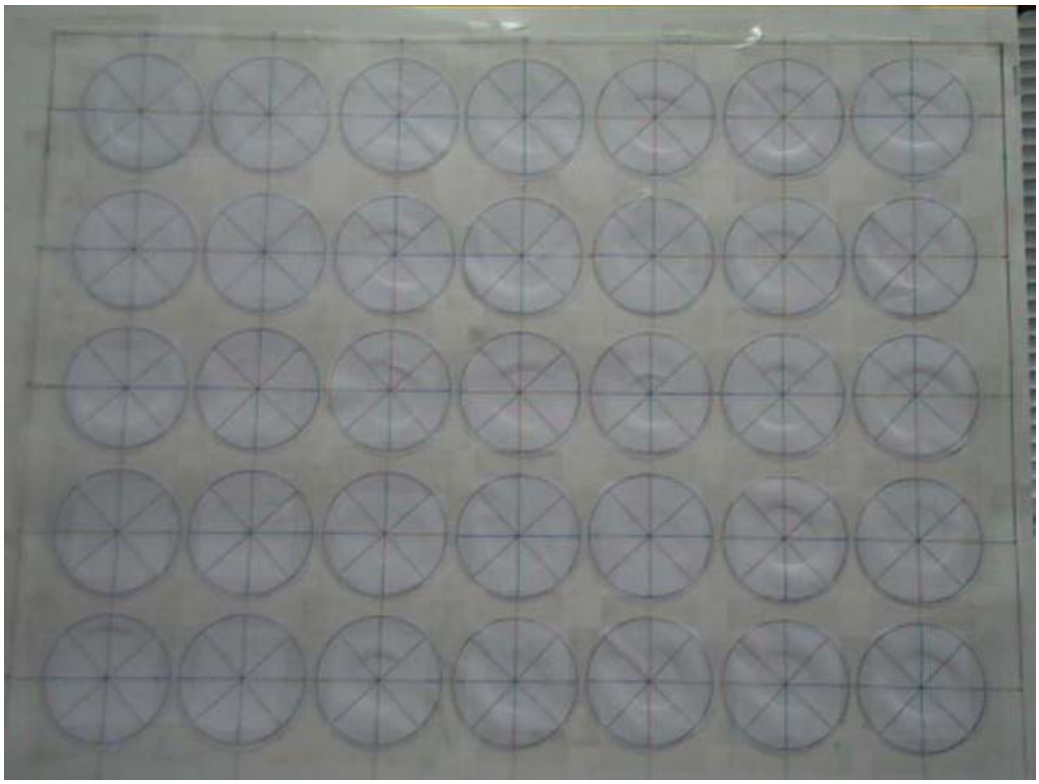

Figure 3.13: Properly aligned target image used for distortion determination

3.4.3 Long Range Calibration

3.4.3.1 Experimental Setup

For the long range camera calibration experiment, the camera was placed in a fixed position atop an adjustable tripod. The pan/tilt device was bolted to an aluminum brace which was in turn fastened to a piece of Unistrut channel adjacent to the camera. This channel was then secured to a metal post. The camera and the pan/tilt device were

placed at the same height and 3.61 m from the wall. A laser pointer was clamped onto the payload platform of the pan/tilt device. The controller, computer, and power supply were placed on a table behind the camera and the pan/tilt device. Figure 3.14 shows a picture of the camera and pan/tilt device in front of a target wall.

Figure 3.14: Experimental Setup for long range calibration. Note that the camera is separated from the pan/tilt device.

The target image was an array of pushpins in the wall of the laboratory. These pushpins were placed 30.5 cm apart from each other except for the far right column and the top row. These were placed 17.8 cm away from the adjacent column and row, respectively. This was done to ensure the incoming video image was filled with targets. Figure 3.15 shows this array of pushpins.

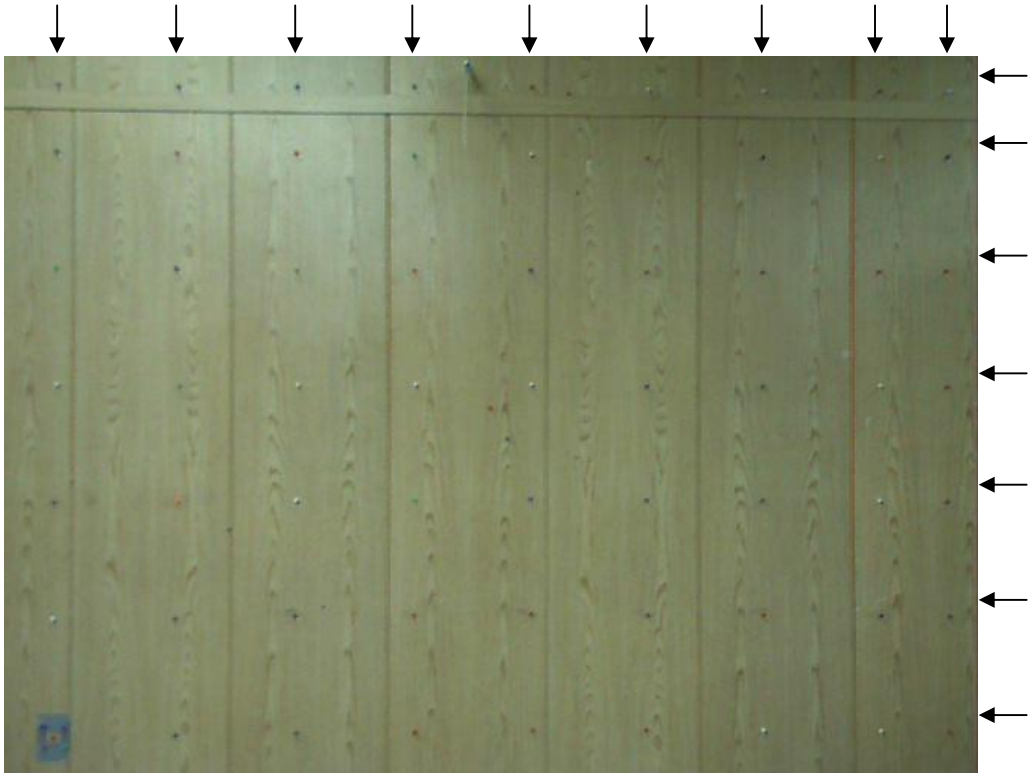

Figure 3.15: Pushpin array (arrows indicate columns and rows)

3.4.3.2 Experimental Procedure

The procedure used to measure the distortion at long range was slightly different from that of the short range calibration. Since the camera was placed atop a tripod, the camera had to be moved manually. Since the exact motion of the camera was not directly measurable when it was on top of the tripod, the angles had to be calculated. To accomplish this, the camera was moved so that the pushpin array was approximately centered in the image. Using the image tool, the center pixel was determined. A pushpin was placed at this location on the wall so that the center of the pushpin was represented by the center pixel of the image. After the camera was moved to a new location, the

image tool was used to find the pixel value of the center of the pushpin. By converting the pixel length into a distance and knowing the distance of the wall from the camera, the degrees of motion of the camera were determined using the following equation:

$$\theta_{P,T} = \tan^{-1}\left(\frac{\Delta P * C}{W}\right) \tag{3-6}$$

where $\theta_{P,T}$ are the degrees of camera motion in either pan or tilt, ΔP is the change in pixel location, W is the perpendicular distance to the wall, and C is the ratio of pixels to distance.

In the short range calibration, minimum pixel length was used to determine the alignment of the camera. For the long range calibration, minimum amount of error shown by the system was used instead of pixel length. The static targeting program was used to aim the laser pointer at each pushpin. The error of the system was evaluated as the sum of the vertical and horizontal distances from the center of the laser pointer to the hole left in the wall by the pushpin. As stated previously, the system calculates position commands from a parameter that relates pixel length to distance. Since the pixel lengths between the pushpins will increase on the image used to aim the pan/tilt device, the error in the system will increase as the target image deviates from parallel alignment with the camera.

Once the camera was aligned, the distortion in the image had to be measured. The pushpins in the wall were positioned to form several squares. The perimeters of these squares were then measured and used in the same way as the diameter of the circles from the previous experiment. The pixel distances that represented the .178 meter distance of the top row and right column were multiplied by a proportion so that they

could be compared to the pixels representing 0.305 meters. The following figures show the vertical and horizontal measurements of the pixel distances between the pushpins (Figures 3.16a and 3.16b). Perimeters were used since squares were much easier to make on the wall than circles.

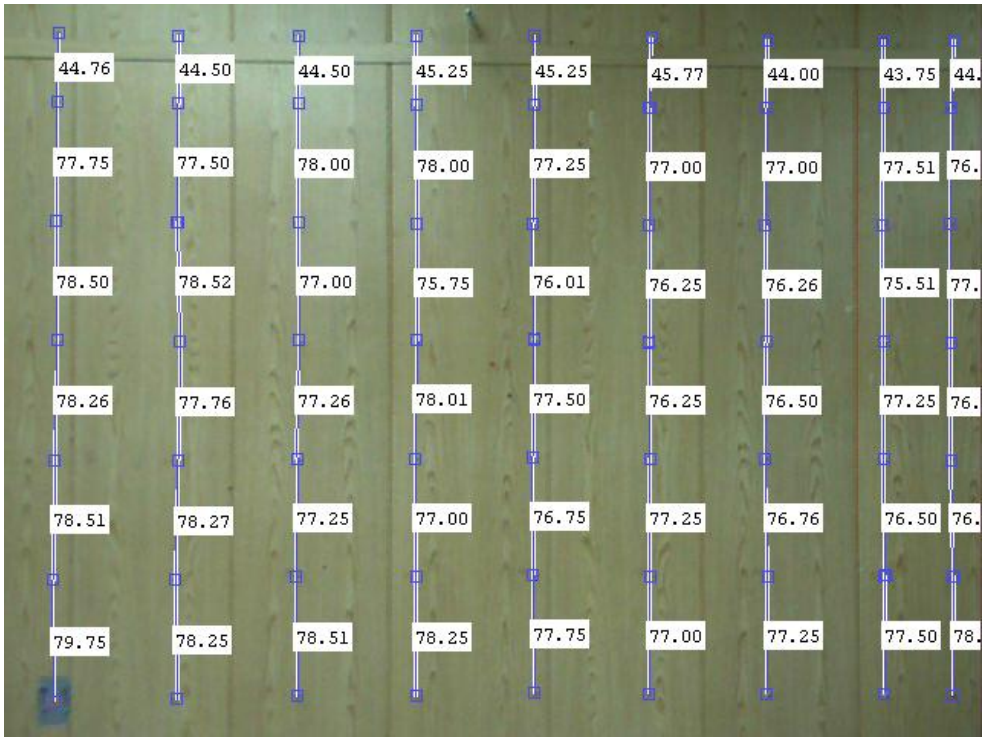

Figure 3.16a: Vertical measurements of the pixel distances between pushpins

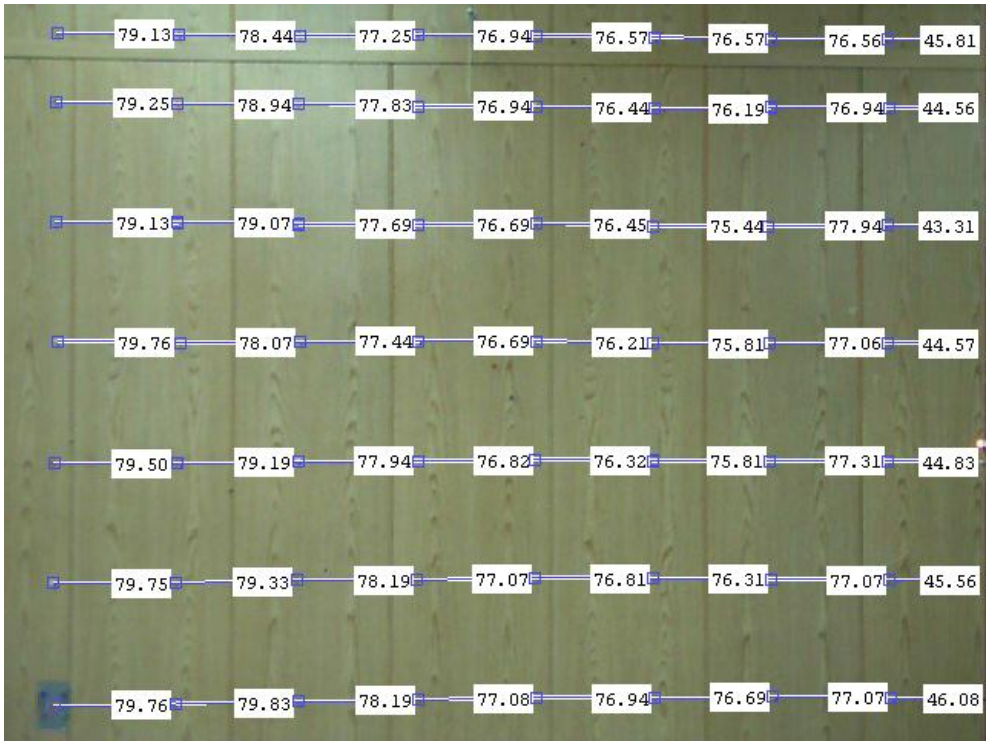

Figure 3.16b: Horizontal measurements of the pixel distances between pushpins

3.4.4 Results and Discussion

3.4.4.1 Camera Alignment

3.4.4.1.1 Short Range Alignment

The center circle of the calibration image was tilted from the top of the image to the bottom of the image at a constant pan angle. The length of the horizontal and vertical diameters was then measured and plotted versus the tilt degree. Four trials were run, and a quadratic trend line was fit to each data set. For each trend line, the minimum pixel length, i.e. the location of zero slope, was determined. These lengths were then averaged

together to give a single angle to be used for the alignment determination. Figures 3.17a and 3.17b show the change in tilt degrees versus the change in pixel length for the vertical and horizontal diameters, respectively. The Y-axis crosses the X-axis at the angle where the average minimum value was calculated (in order to aid the reader in seeing the minimum value).

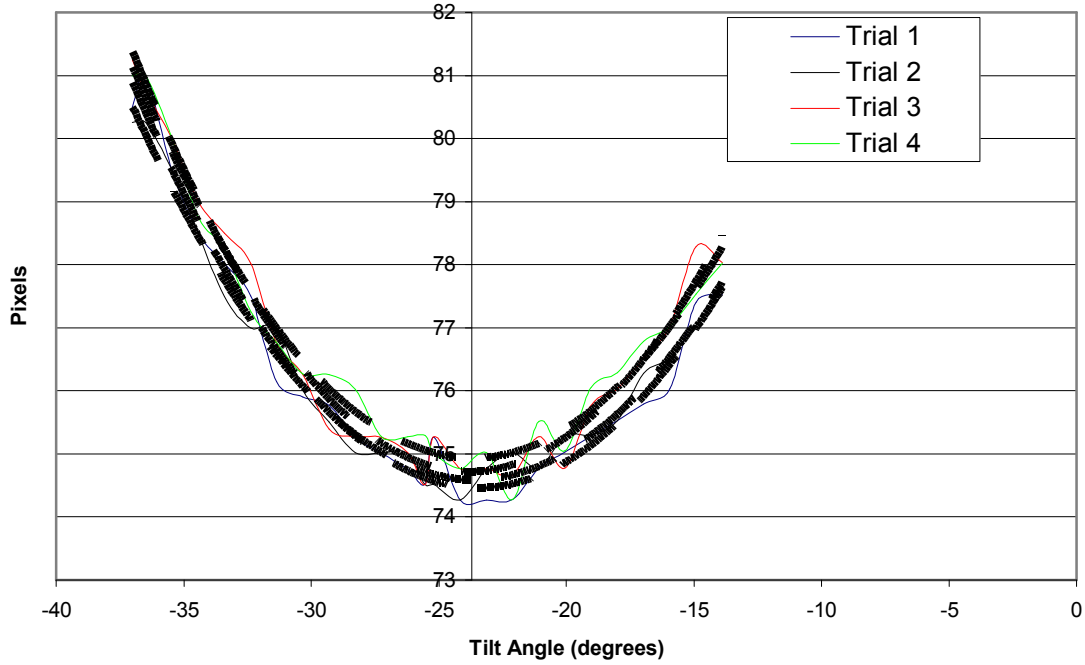

Figure 3.17a: Change in the pixel length of the vertical diameter as the camera was tilted at a constant pan angle of 0.6682° to the right

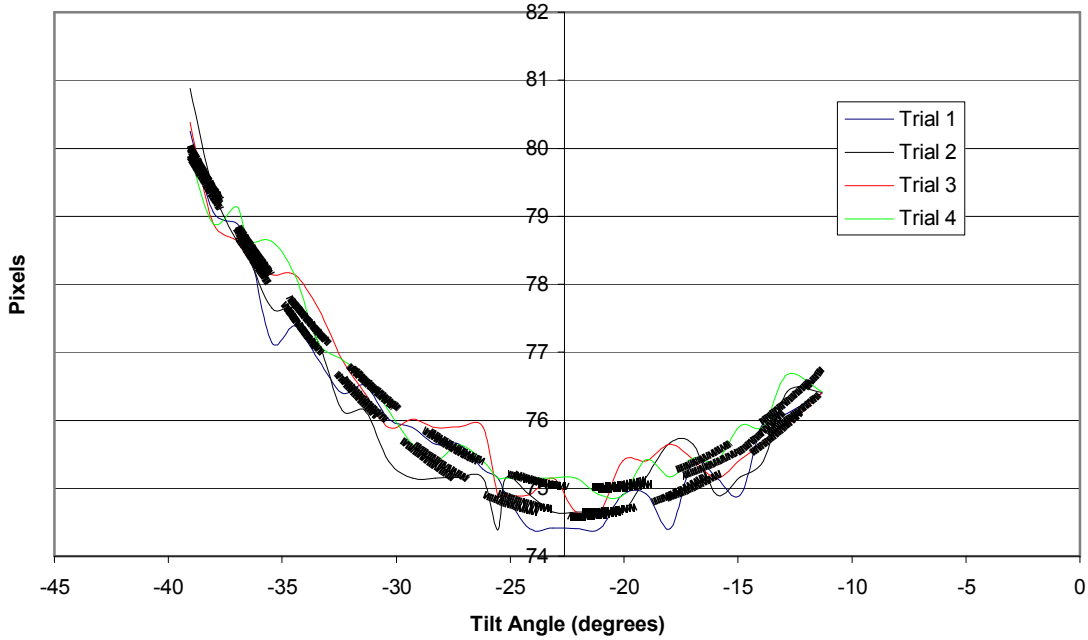

Figure 3.17b: Change in the pixel length of the horizontal diameter as the camera was tilted at a constant pan angle of 0.6682° to the right

This process was repeated by moving the camera horizontally while keeping the angle of tilt constant. Figures 3.18a and 3.18b show the change in pan angle versus the change in pixel length for the vertical and horizontal diameters, respectively. Again, the Y-axis crosses the X-axis at the degree where the average minimum value was calculated.

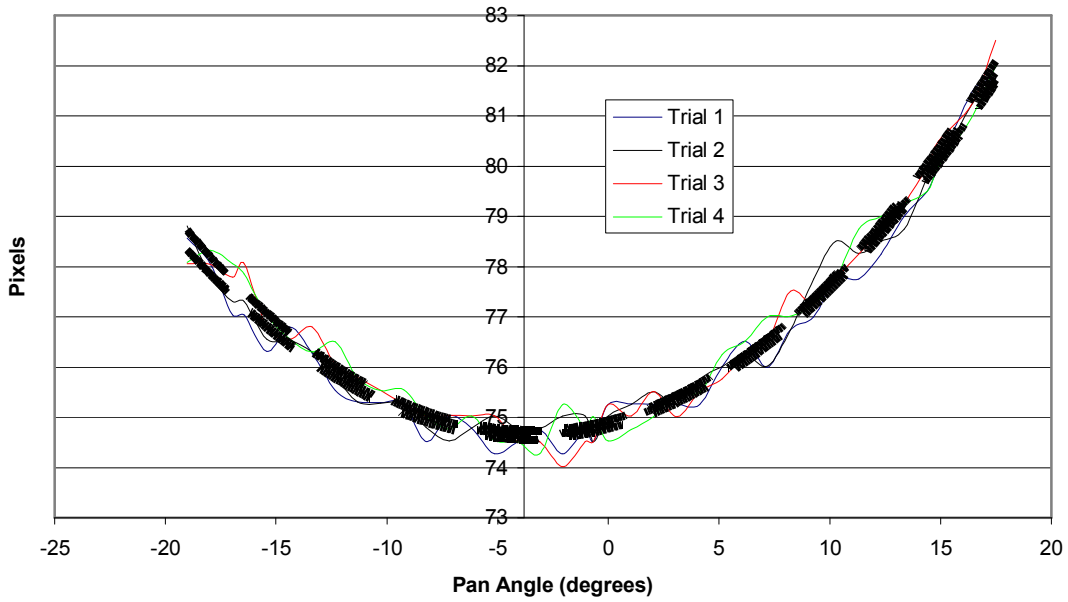

Figure 3.18a: Change in the pixel length of the vertical diameter as the camera was panned at a constant tilt angle of 25.55° down

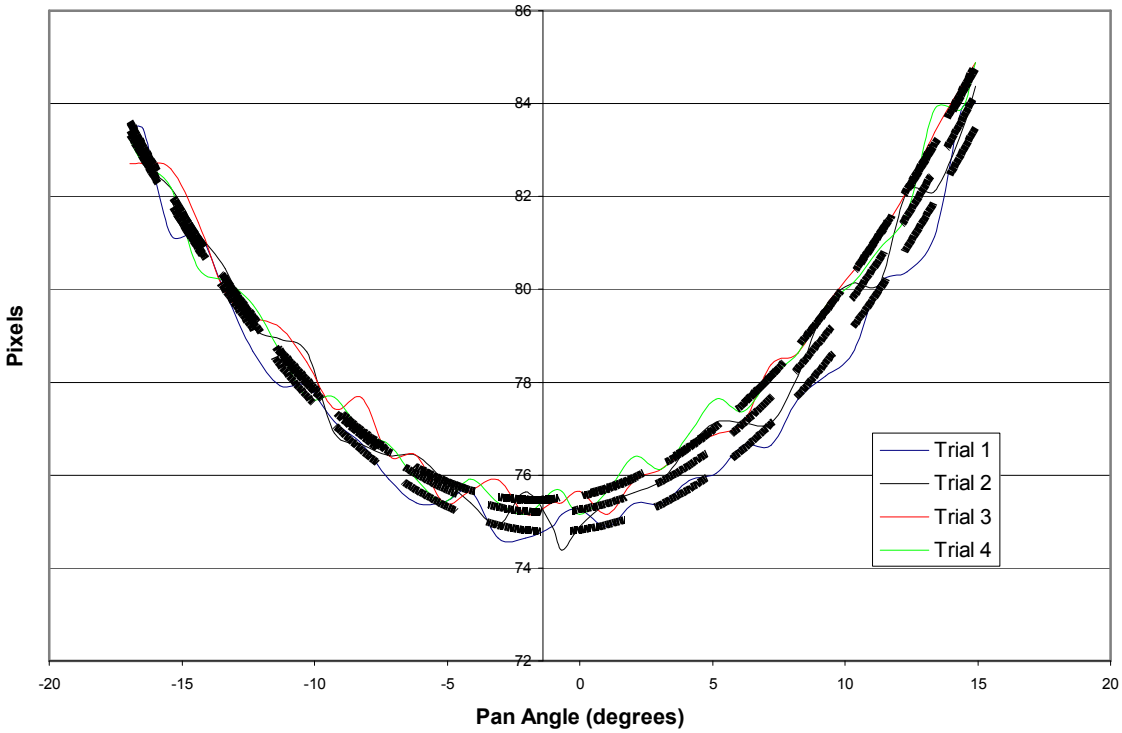

Figure 3.18b: Change in the pixel length of the horizontal diameter as the camera was panned at a constant tilt angle of 25.55° down

From these data, four angles of camera motion were evaluated (two for pan and two for tilt). The angles of minimum pixel value were then matched according to diameter, e.g. the pan angle determined from using the horizontal diameter while panning at a constant tilt angle was combined with the tilt angle determined from using the horizontal diameter while tilting at a constant pan angle. Combining these results created two sets of pan and tilt angles.

In order to obtain more data to more confidently assess camera alignment, the alignment circle was also moved in a diagonal pattern across the image from the upper right corner to the lower left corner and from the upper left corner to the lower right corner. The change in tilt angle was plotted versus the change in pixel lengths. On the same graph, pan angle was plotted versus tilt angle on a second Y-axis. This was done for both diameters creating four graphs.

Four trials were done where the change in tilt angle was plotted versus the change in pixel length. Parabolic trend lines were fit to this data to find the angle of tilt where the pixel length was a minimum. The Y-axis for the pan angle was moved to intersect the X-axis at the location of average minimum pixel length. Like the previous graphs, this was done to aid in the determination of the minimum pixel value.

Once the tilt angle associated with the minimum pixel length was found, the pan angle where this occurred had to be determined. To get this pan angle, the pan angle of camera motion was plotted versus the tilt angle of camera motion. This represented the physical motion of the camera as it moved from one corner to the next. A linear trend line was fit to this data. The pan angle associated with the tilt angle of minimum pixel value was determined by taking the intercept of the trend line and the pan angle Y axis. Figures 3.19a and 3.19b show the changes for both diameters as the circle move from the upper right corner to the lower left corner. In order to aid the understanding of these graphs, it may be helpful to think of the pixels versus tilt angle portion as the determination of the minimum value and the pan angle versus tilt angle portion as a reference to get the combination of pan and tilt degrees necessary to reach that value.

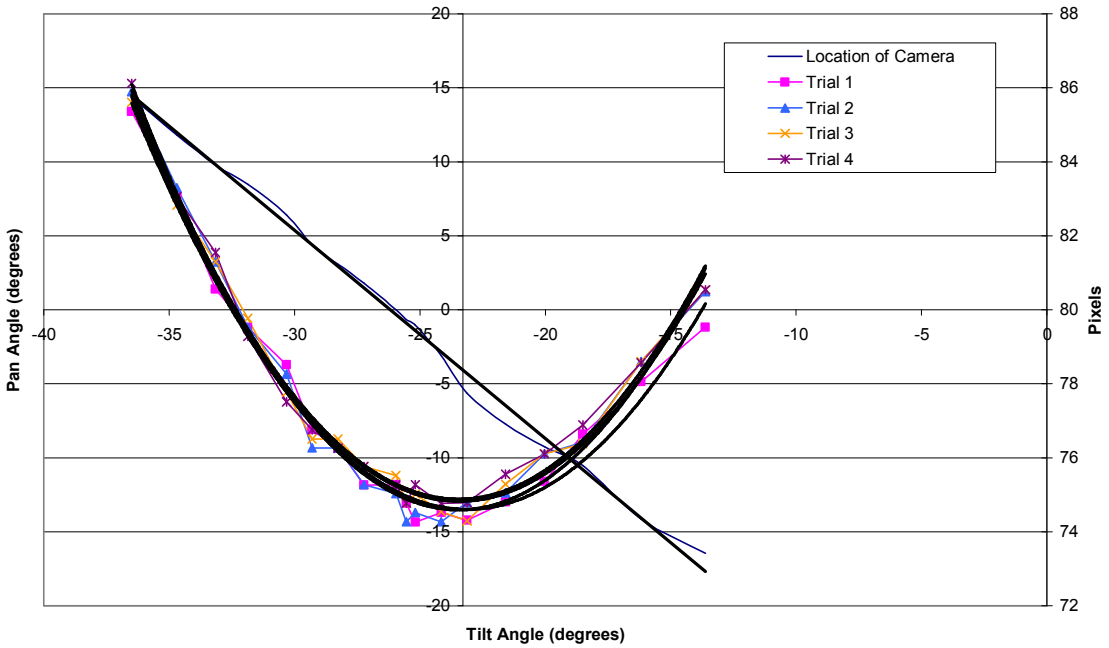

Figure 3.19a: Change in the pixel length of the vertical diameter as the camera was moved diagonally from the upper right corner of the image to the lower left corner

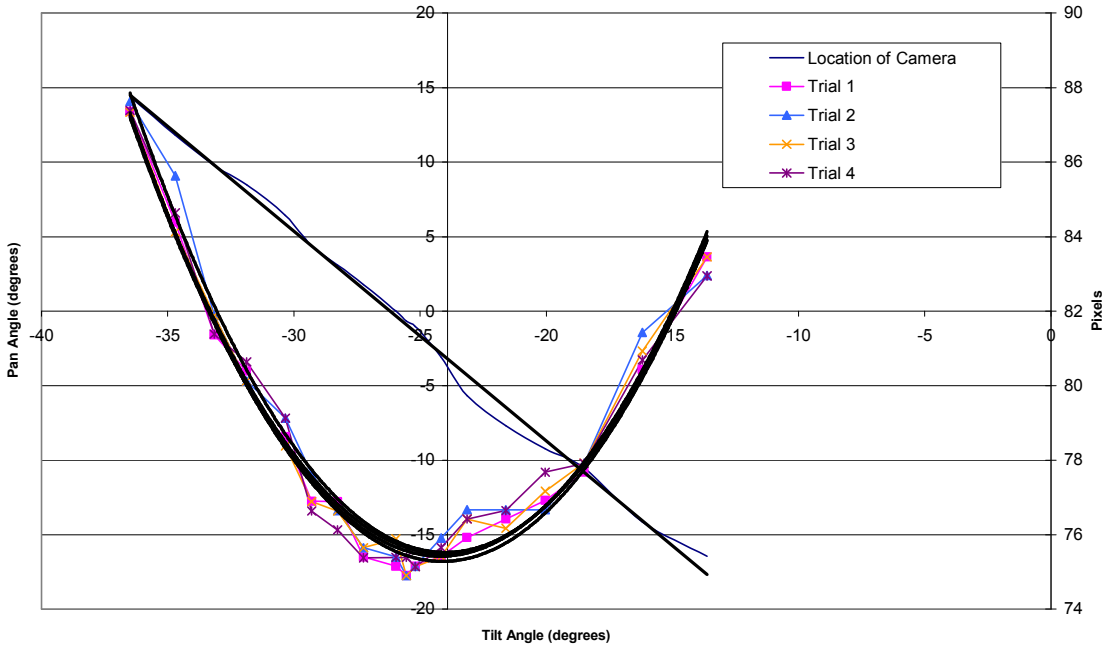

Figure 3.19b: Change in the pixel length of the horizontal diameter as the camera was moved diagonally from the upper right corner of the image to the lower left corner

Figures 3.20a and 3.20b show the changes for both diameters as the circle moved from the upper left corner of the image to the lower right corner.

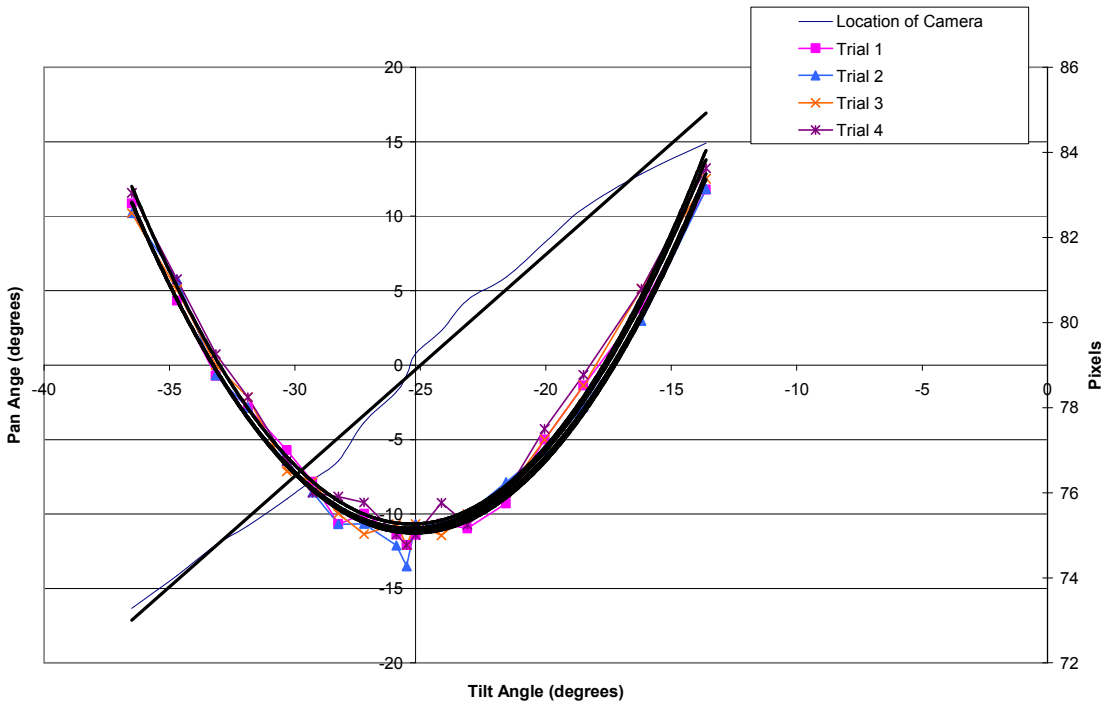

Figure 3.20a: Change in the pixel length of the vertical diameter as the camera was moved diagonally from the upper left corner of the image to the lower right corner

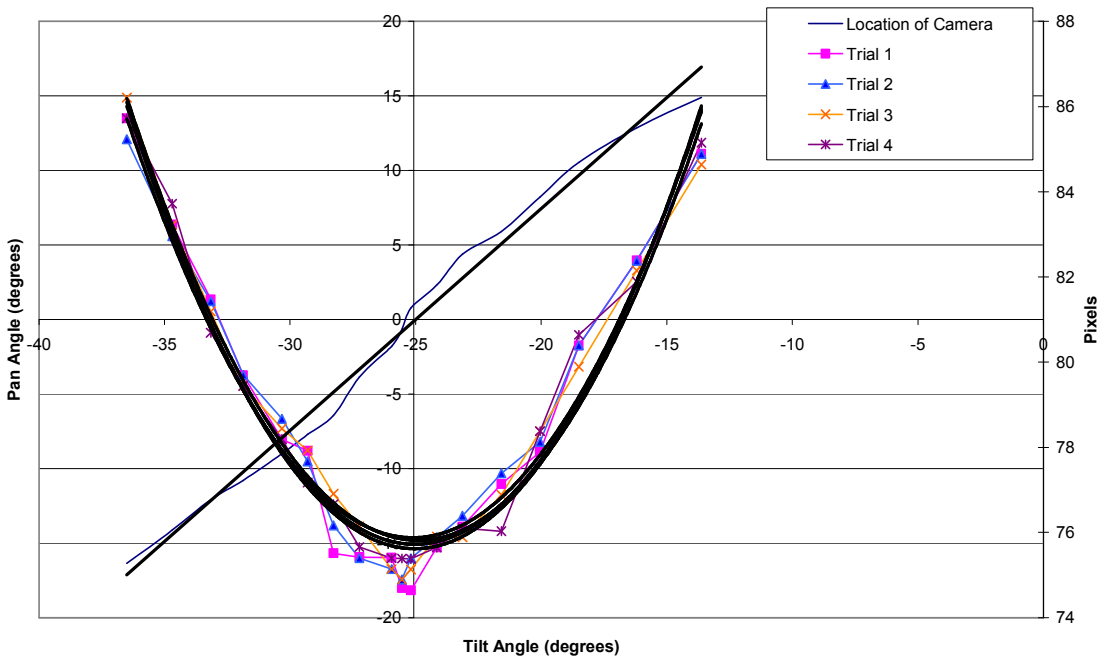

Figure 3.20b: Change in the pixel length of the horizontal diameter as the camera was moved diagonally from the upper left corner of the image to the lower right corner

From these graphs, four more sets of pan and tilt pairs were obtained. These four were combined with the two obtained from the previous trials where motion in one direction was held constant. These six sets were average together to get the pan and tilt degrees that the pan/tilt device would have to move in order to align the image sensor in the camera with the target image. Figure 3.21 shows the results of each trial as well as the average value.

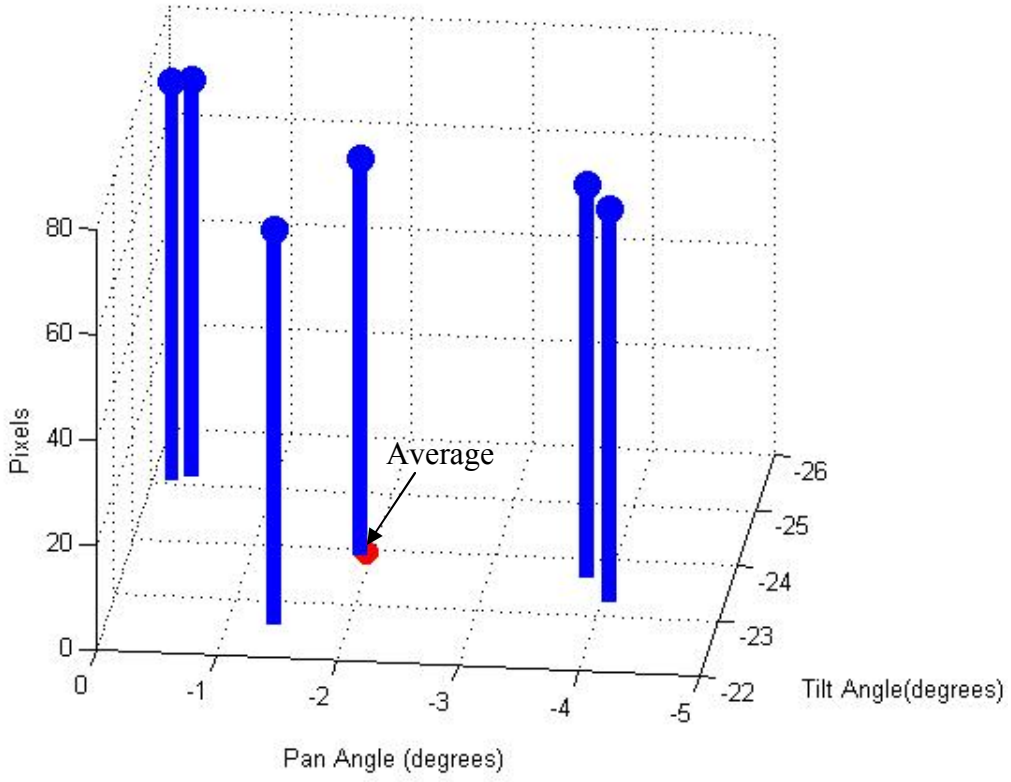

Figure 3.21: The locations of the minimum pixel values for each trial as well as the average of the 6 sets of pairs obtained from the trials

3.4.4.1.2 Long Range Alignment

The camera alignment and distortion calibration was repeated for the longer range experimental configuration. Since most of the image contained pushpins, the actual motion of the camera was very limited. This is because all of the pushpins needed to be in the image to measure the total system error. The matrix was initially in the center of the image. After the error was found, the camera was moved until the minimum error was found. Those data are shown in Table 3.1.

Table 3.1: Total system error measured during long range camera alignment

Position of the Camera	Total System Error
0° pan, 0° tilt	43.75 inches
.036° left from center, 0° tilt	38.12 inches
.018° right from center, 0° tilt	42.875 inches
0° pan, .12° down from center	53 inches
0° pan, .22° up from center	33.125 inches
.026° right from center, .22° up from center	34.68 inches
.038° left from center, .22° up from center	30.19 inches

As can be seen from the tabulated data, moving the camera left and up from the center position produced the smallest error. However, in order to ensure that the camera was aligned, the camera was panned and tilted further. Since this eliminated some of the pushpins from the image, only the remaining pushpins were measured to determine the system error. These were compared with the same pushpins that had previously been measured. The errors were then plotted versus motion from the previous best position that was determined from the data in Table 3.2. Figures 3.22a and 3.22b show the change in total error as the camera was tilted and panned, respectively.

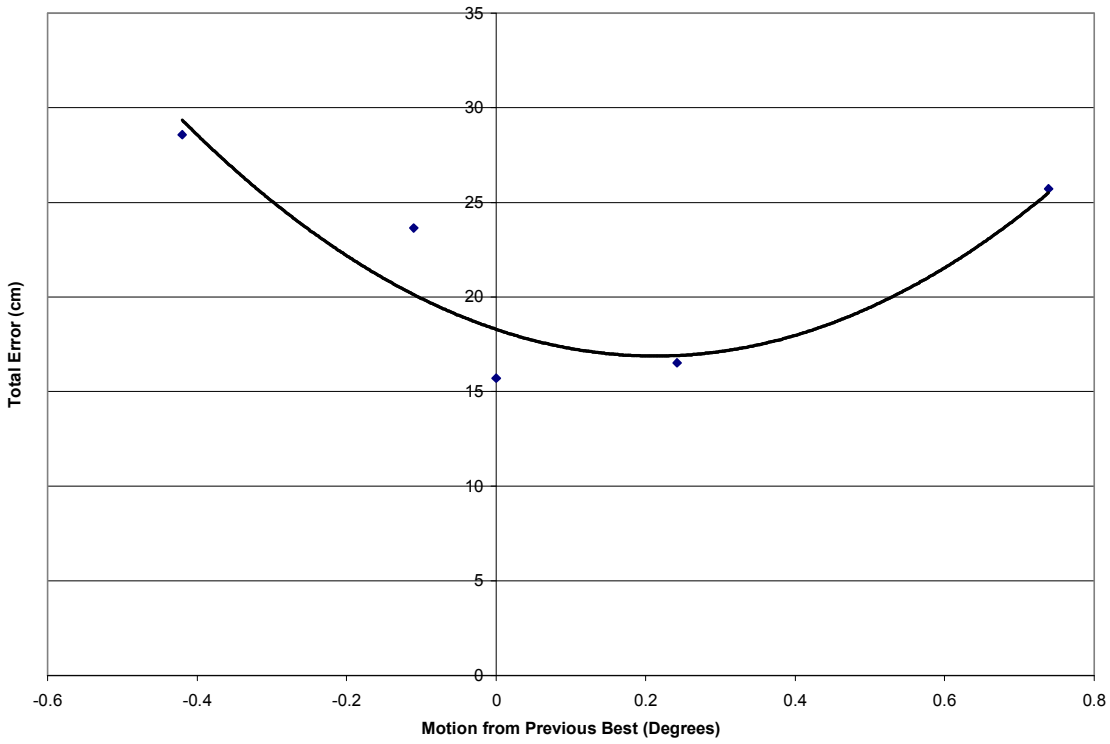

Figure 3.22a: Change in error as the camera is tilted up and down from the previous best location

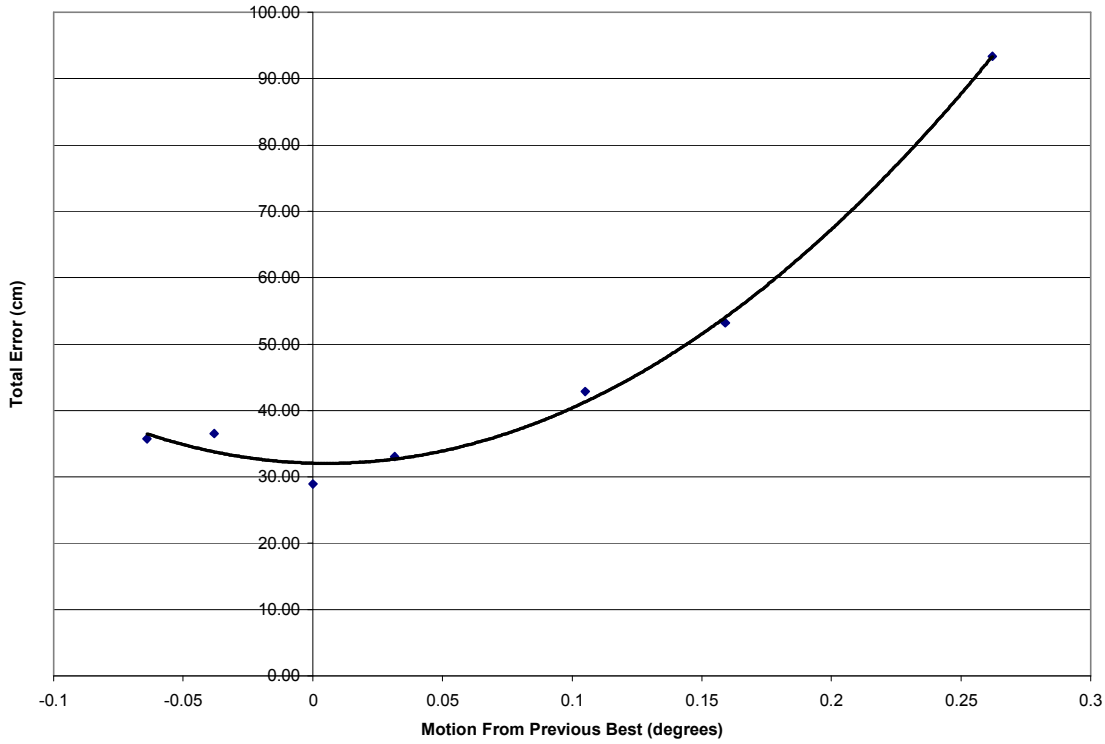

Figure 3.22b: Change in error as the camera is panned left and right from the previous best location

Trend lines were fit to both graphs to ensure that the error increased as it moved away from the best location in any direction. As can be seen, this is the case.

3.4.4.2 Camera Calibration

3.4.4.2.1 Short Range Calibration

Once the camera was properly aligned with the target image, the camera lens distortion was reassessed. Recall that the calibration image was a poster containing a matrix of identical circles. Each circle was measured across its diameter in four locations

and averaged together. The average from the center circle was used to normalize the other circles. This was accomplished by dividing all the averages by the center circle's average. By doing this, a table was constructed to show how the sizes of the circles change throughout the image. This table (Table 3.2) is shown below. The statistical data is ± 2σ for each corner.

Table 3.2: Deviation of circle sizes in the image using the center circle as a reference

± .006						± .004
.99	.98	.97	.98	.98	.99	.99
1.01	1.01	1.00	1.00	1.00	.99	1.01
1.03	1.02	1.01	1	1.01	1.02	1.03
1.05	1.04	1.03	1.03	1.04	1.05	1.06
1.08	1.08	1.08	1.07	1.08	1.07	1.09
± .003						± .007

If the image had no distortion at all, all the circles would be the same size and all the values in Table 3.1 would be equal to 1. Since they are not, it can be said with certainty that the lens in the camera contains imperfections giving rise to distortion. As mentioned earlier, there are three common types of image distortion: barrel shaped, pincushion shaped, or a combination of the two [3, 7]. Examples of each type, along with an undistorted image, are given below in Figure 3.23.

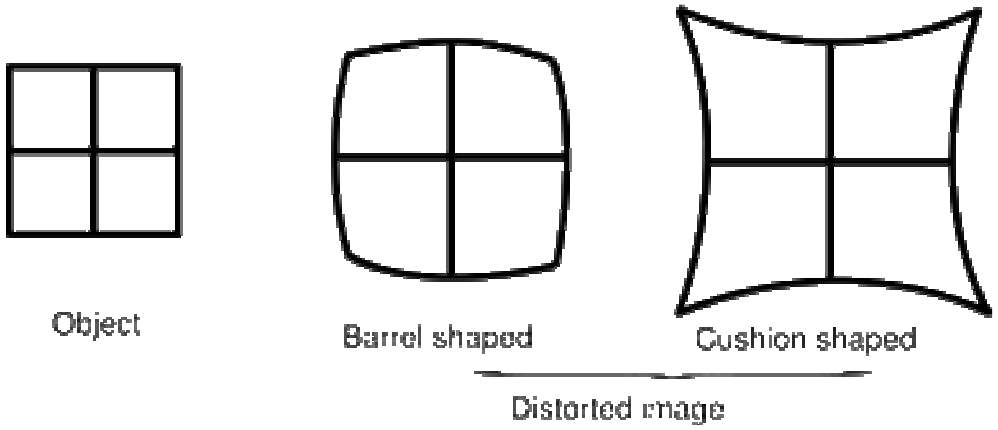

Figure 3.23: The two common types of distortion [9]

The prevailing type of distortion is most easily seen around the perimeter. According to the tabulated data, the lens in this camera has a combination of the two types. Figure 3.24 shows the target image with the prevailing type of distortion on the perimeter indicated.

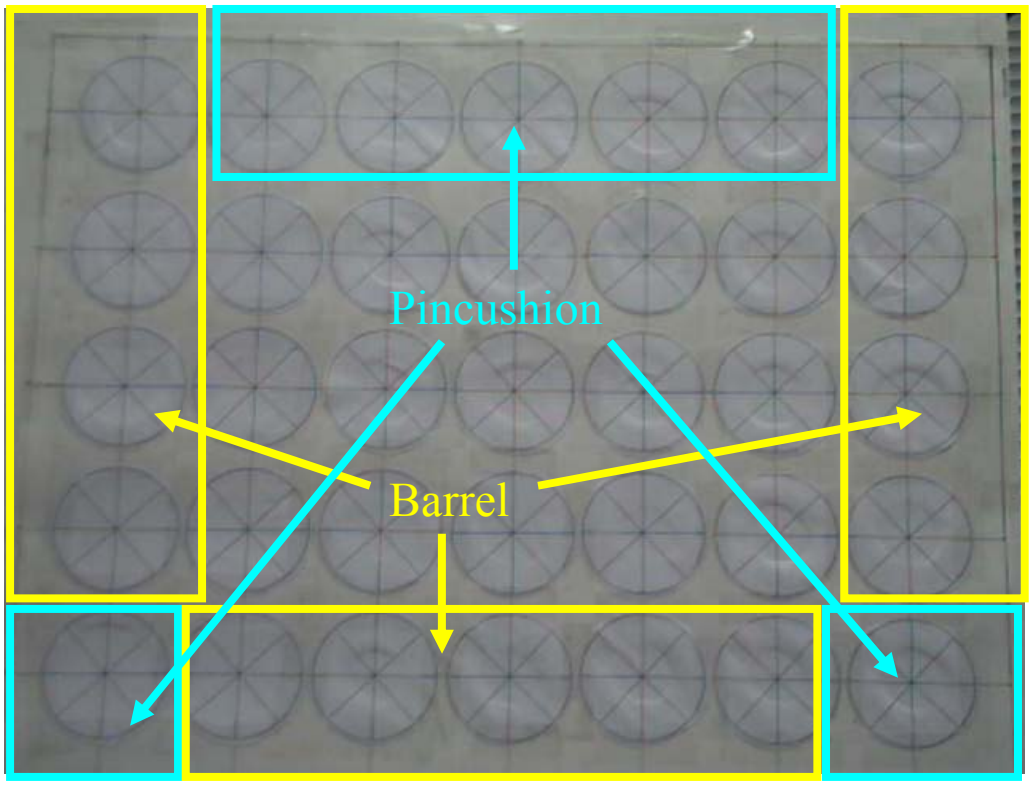

Figure 3.24: The prevailing distortion types on the perimeter of the image

3.4.2.2.2 Long Range Calibration

Since the camera images from the longer range were focused by only part of the camera lens, the camera was recalibrated to ensure minimum error. The targeting program was used to aim the laser pointer at each pushpin. The vertical and horizontal error of the laser spot was measured for each case. The pixel location of each pushpin was also determined. These locations and errors were then placed in a matrix to create a look-up table. A program was then written that would be called when either targeting program was run. By using the same pixel value inputs that the targeting programs use to determine where to aim the pan/tilt device as inputs into the correction program, the

errors associated with that pixel value would be brought into the program. These length errors would then be converted into angular corrections and added or subtracted to the angles calculated by the program. These corrected angles were then sent to the pan/tilt device to accurately aim it at the desired target. For targets in-between the pushpins, a weighted average of the values of the two pushpins to the left and right as well as above and below the identified pixel was used to determine the correction needed to move the pan/tilt device to that location.

4. TESTING OF STATIC TARGETING CAPABILITY

4.1 Purpose and Objective

Once the system was properly aligned and calibrated, its feasibility as a practical system needed to be determined. One of the things to be tested was its static targeting capability. Experiments were run to quantify both the accuracy as well as the performance of the system. However, before this was done, the geometric model used to describe the system and control the pan/tilt device had to be tested as well to ensure its validity.

4.2 Results and Discussion

4.2.1 Hyperbolic Correction

The test to prove the necessity of the hyperbolic correction was easy to perform. The static targeting program was used to aim the laser pointer at the pushpins. The actual position of the laser pointer after it had moved was determined by the geometric model used. The original model was used first; the test was then repeated with the hyperbolic correction added to the original model. By plotting the response of the laser pointer versus the position of the pushpin, it was simple to determine whether the hyperbolic

correction was needed. The model without the hyperbolic correction moved in a curvilinear fashion. The model with the correction was much closer to the actual locations as shown in Figure 4.1. It should be reemphasized that this test was performed before the final image calibration was applied to the system.

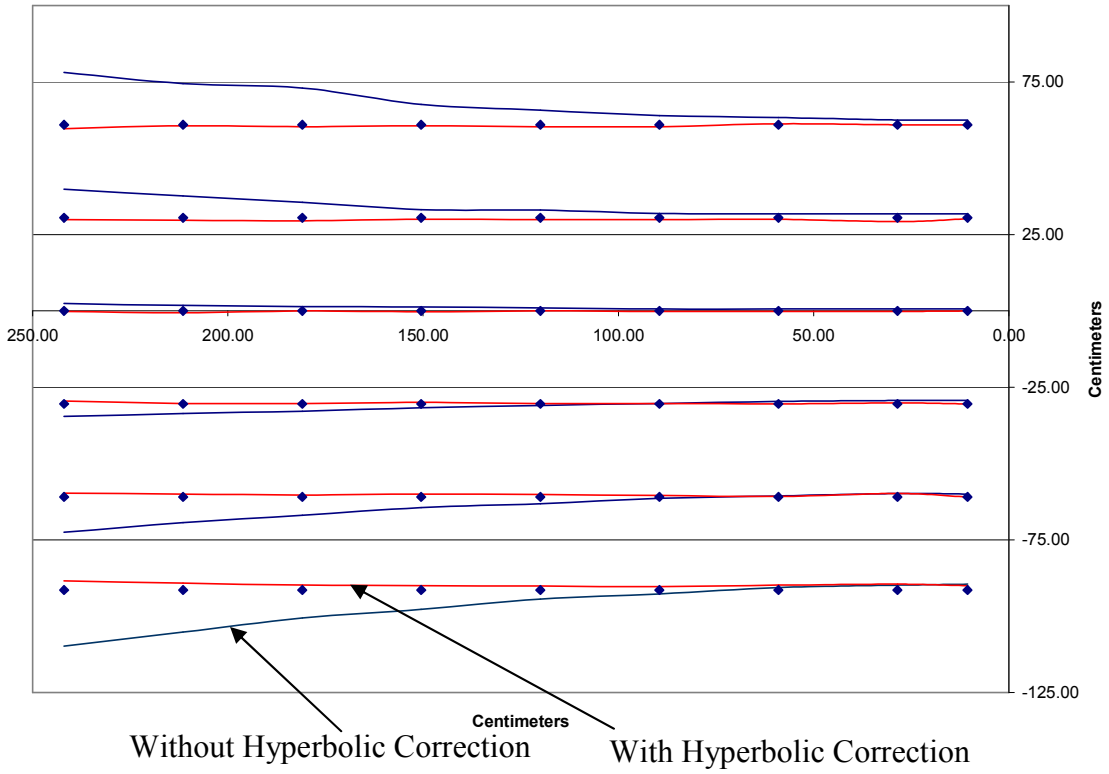

Figure 4.1: Response of the geometric models versus the pushpin locations. Pushpin locations are shown by data symbols.

4.2.2 Complete Model

After the verification of the hyperbolic correction, the complete geometric model was tested. The location of the laser pointer after it was aimed at each pushpin was plotted versus the location of the pushpins. This is shown below in Figure 4.2.

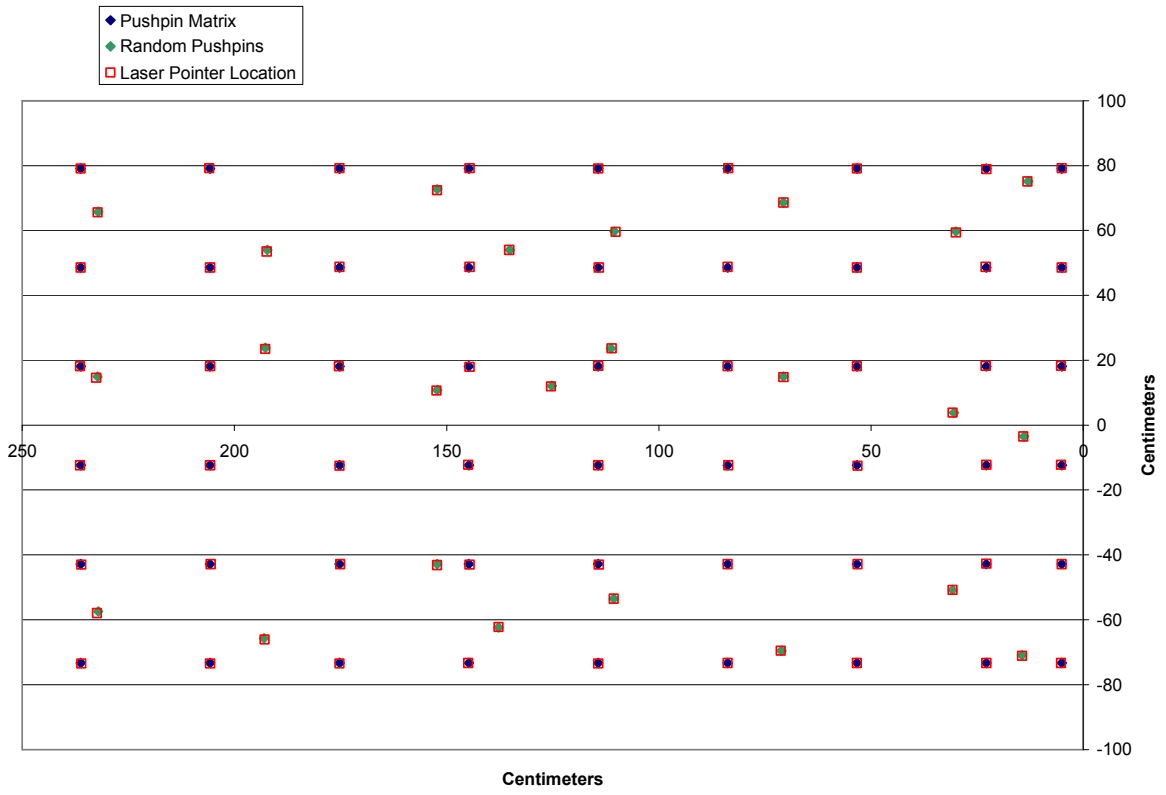

Figure 4.2: Results of geometric model test

From this figure, it can be seen that the geometric model was not 100% accurate. The largest errors were seen on the left hand side where the laser pointer is too high in the lower portion and too low in the higher portion. It then became necessary to explain this.

In order to do that, calculations were done to see how the response changed for various types of misalignment of the pan/tilt device in reference to the geometrical model that described it, e.g. how the response of the system would change if the distance from the pan/tilt device to the target plane was different in reality than it was in the geometric model.

4.2.3 Misalignment Maps

There are six different ways that the system could be misaligned. These six can be grouped into two categories: translational misalignments and rotational misalignments. The translational misalignments occur when either the pan/tilt device is at a different distance from the wall than the distance used in the model or when the pushpins are not at the correct positions form the origin, i.e. the origin is shifted horizontally or vertically. These can be thought of as a translation along an axis in a three dimensional space. The rotational misalignments occur when the pan/tilt system is not perfectly perpendicular with the target plane. An example of the three rotations is given in Figure 4.3.

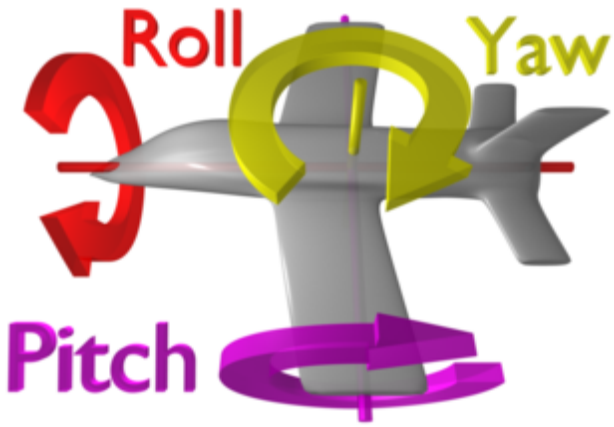

Figure 4.3: The three types of rotational misalignment that the pan/tilt device can experience [27]

By varying the model in these six ways, six misalignment maps were created. For the translational maps, a translation of ± 1.27 cm was added to the model along each axis. The results are given below as an overall effect and a close up of the bottom row of pushpins since some of the variations are small (Figures 4.4, 4.5, 4.6, 4.7, 4.8, and 4.9). The legend for each pair of figures is shown in the close up.

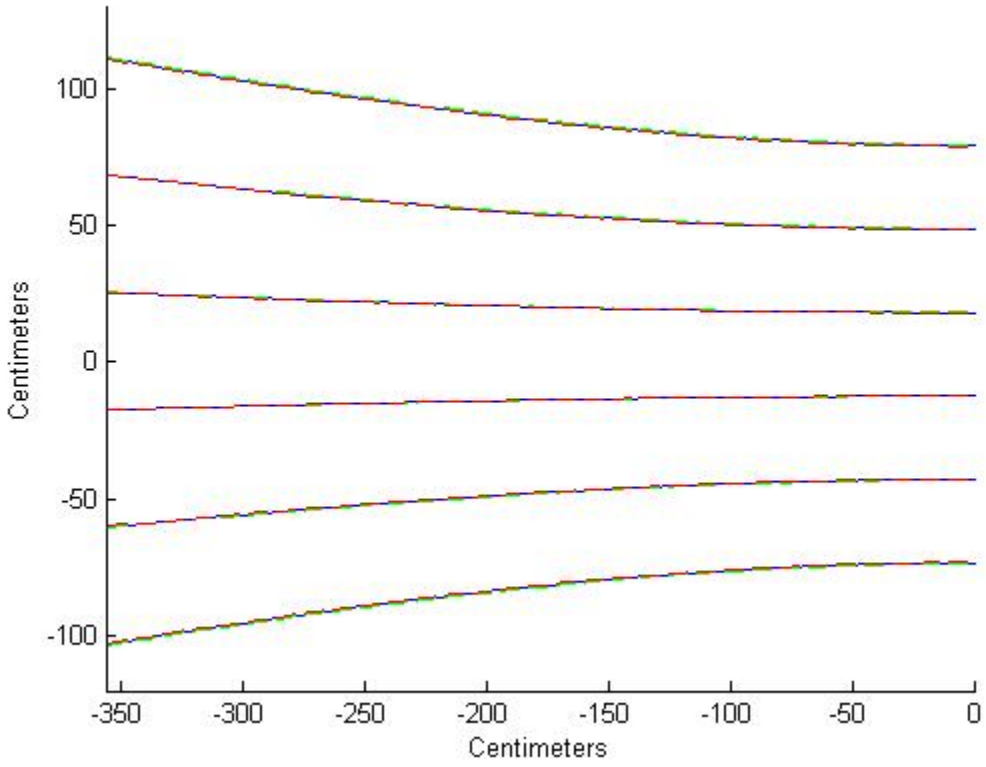

Figure 4.4: The variation in the hyperbolic correction from 1.27 cm motion of the pan/tilt device in relation to distance from the wall

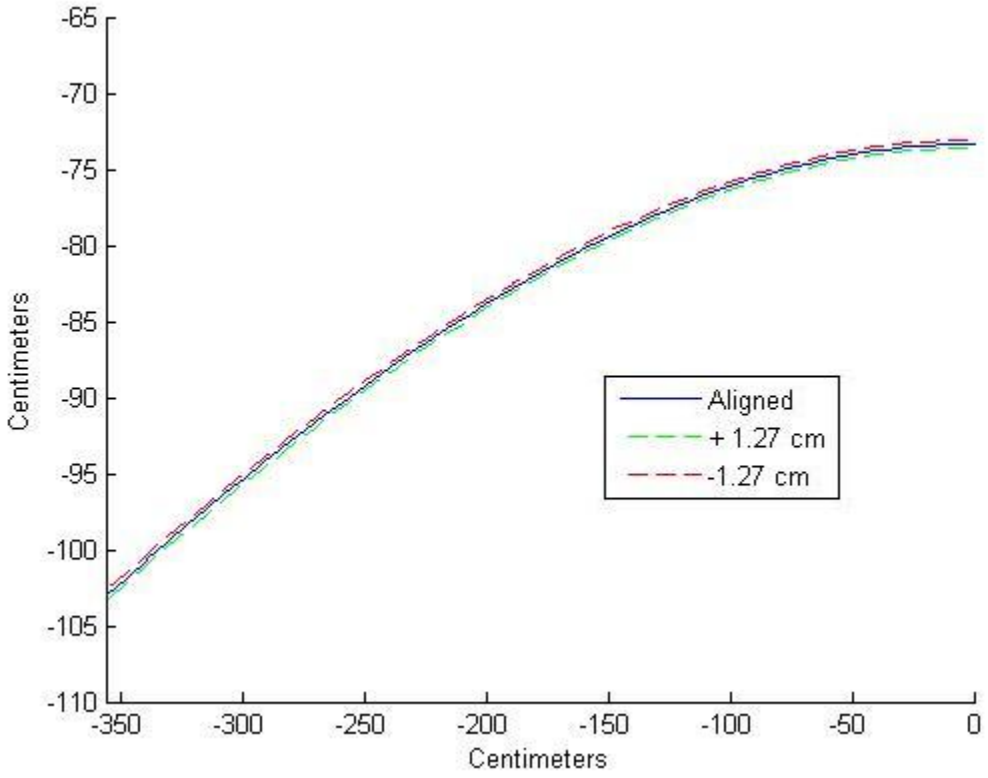

Figure 4.5: The variation in the hyperbolic correction from 1.27 cm motion of the pan/tilt device in relation to distance from the wall for the bottom row of pushpins

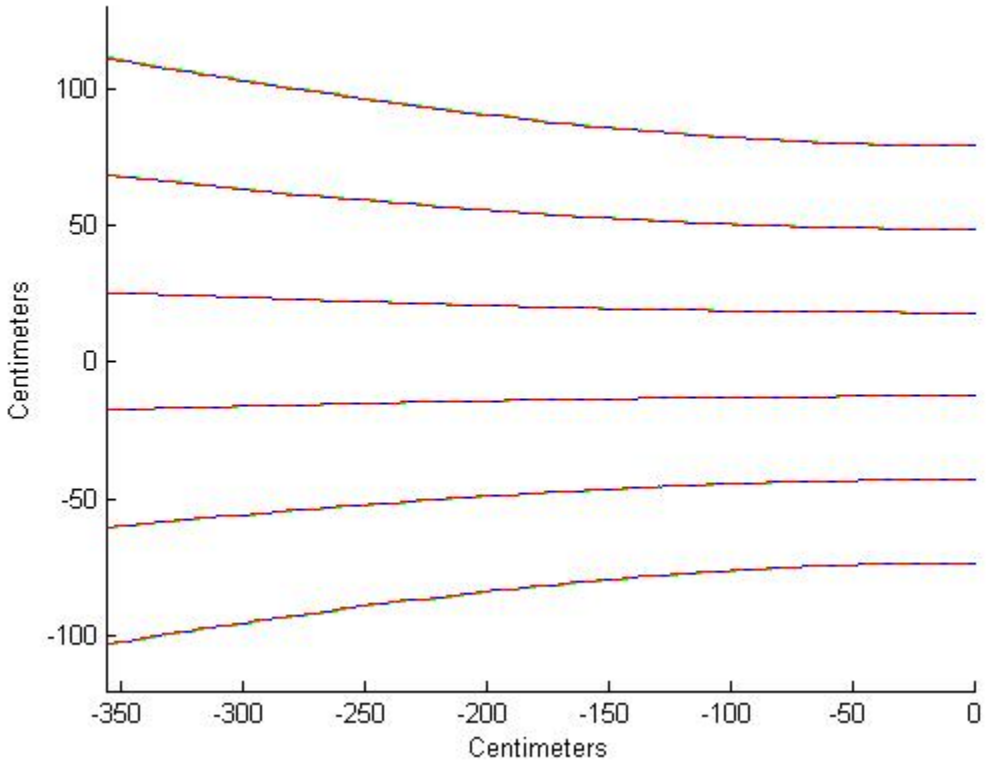

Figure 4.6: The variation in the hyperbolic correction from 1.27 cm horizontal motion of the pan/tilt device in relation to the origin

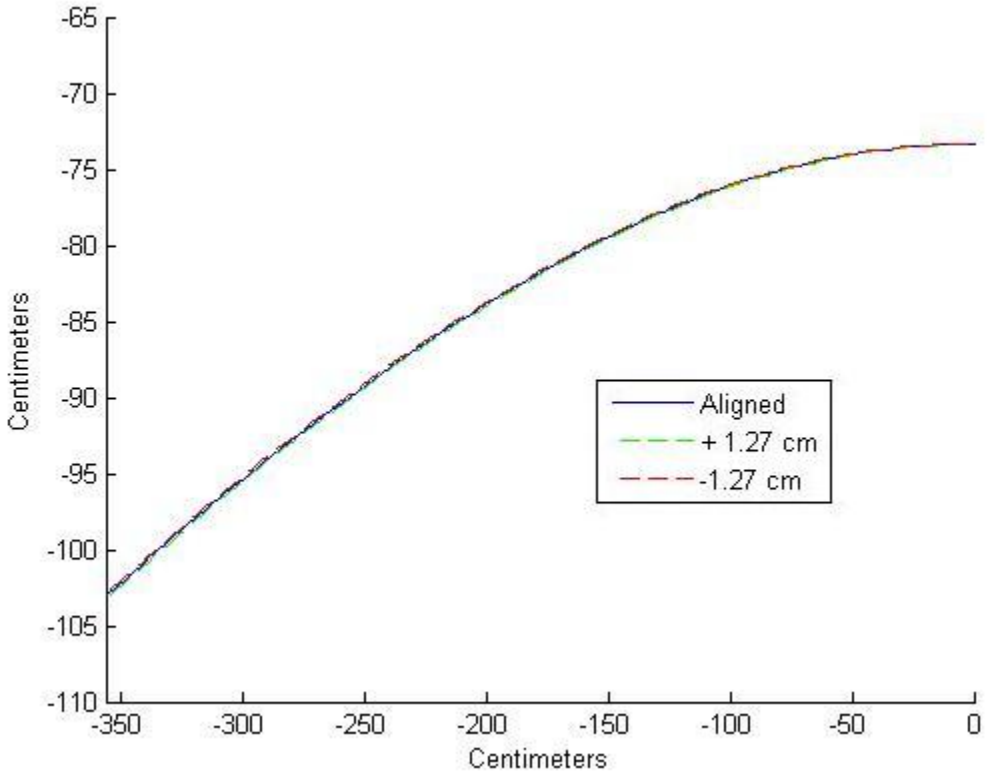

Figure 4.7: The variation in the hyperbolic correction from 1.27 cm horizontal motion of the pan/tilt device in relation to the origin for the bottom row of pushpins

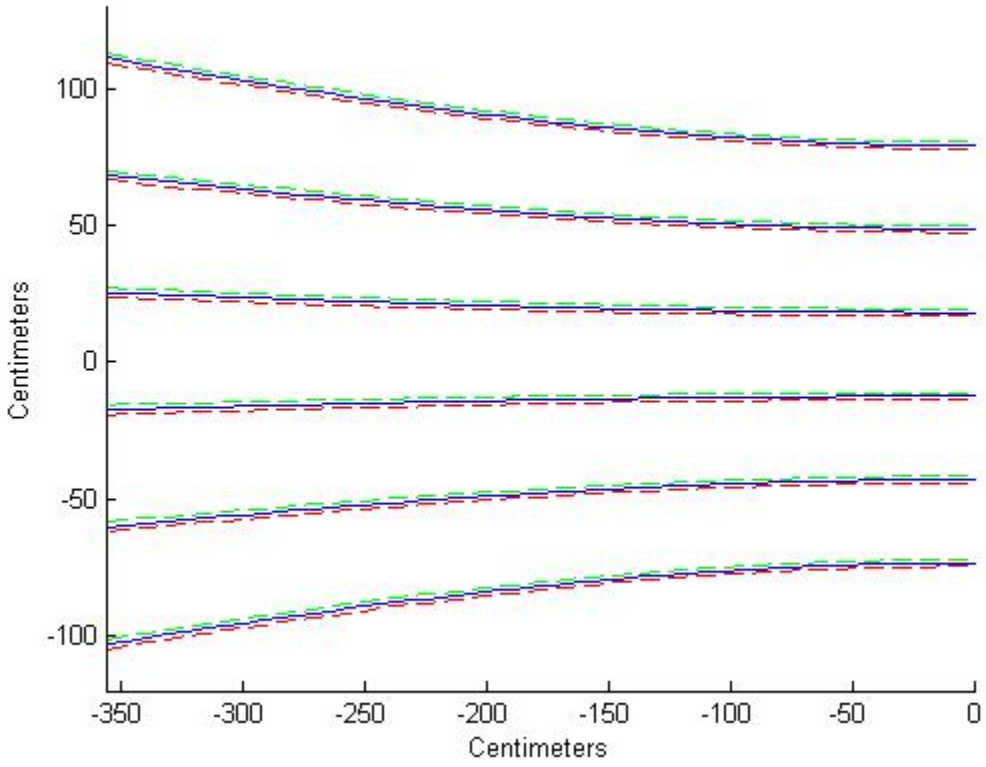

Figure 4.8: The variation in the hyperbolic correction from 1.27 cm vertical motion of the pan/tilt device in relation to the origin

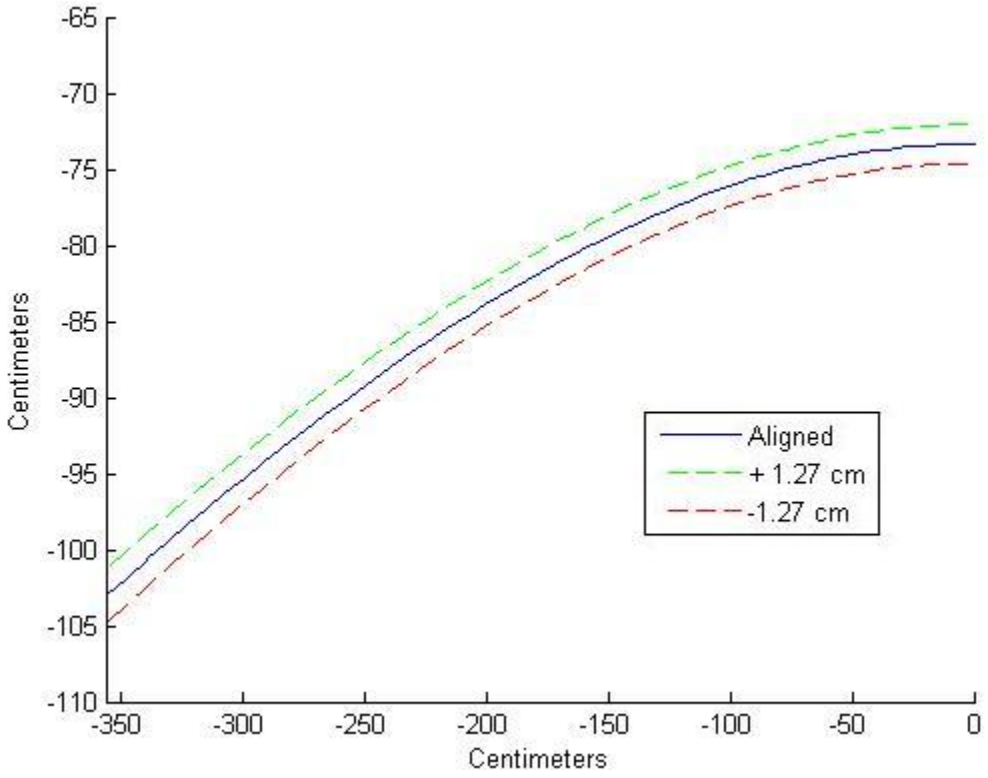

Figure 4.9: The variation in the hyperbolic correction from 1.27 cm vertical motion of the pan/tilt device in relation to the origin for the bottom row of pushpins

For the rotational maps, a rotation of ± 1° around each axis was added to the model. The results are given below as an overall effect and a close up of the bottom row of pushpins since some of the variations are small (Figures 4.10, 4.11, 4.12, 4.13, 4.14, and 4.15). Again, the legend for each pair of figures is shown in the close up.

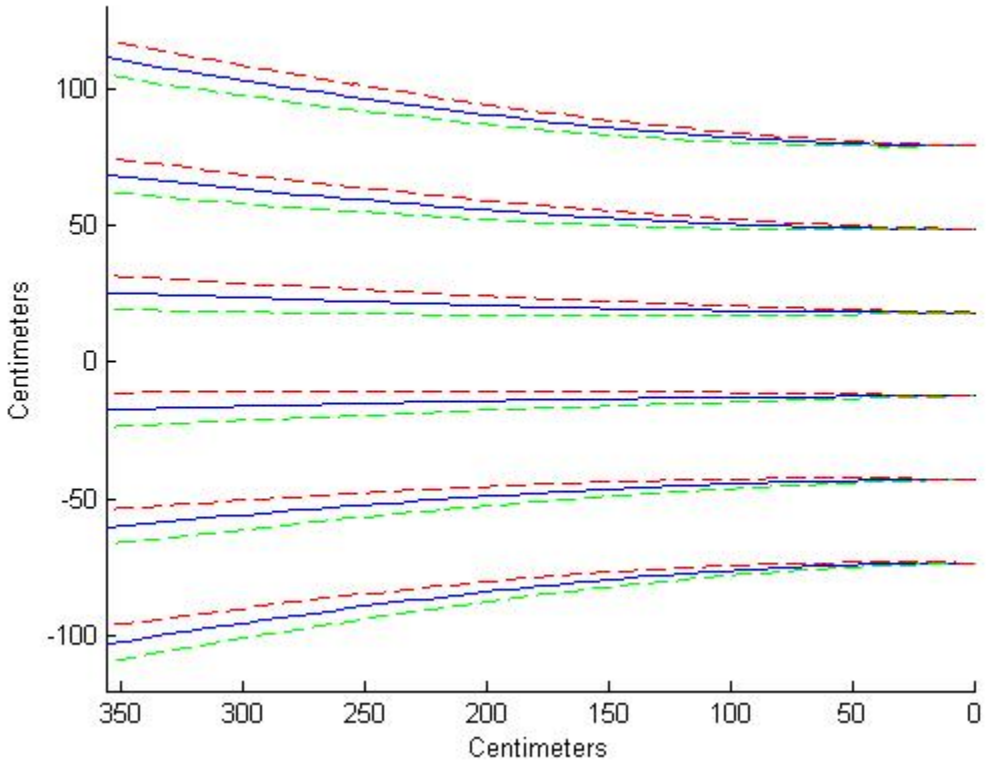

Figure 4.10: The variation in the hyperbolic correction from 1° misalignment of the pan/tilt device in roll

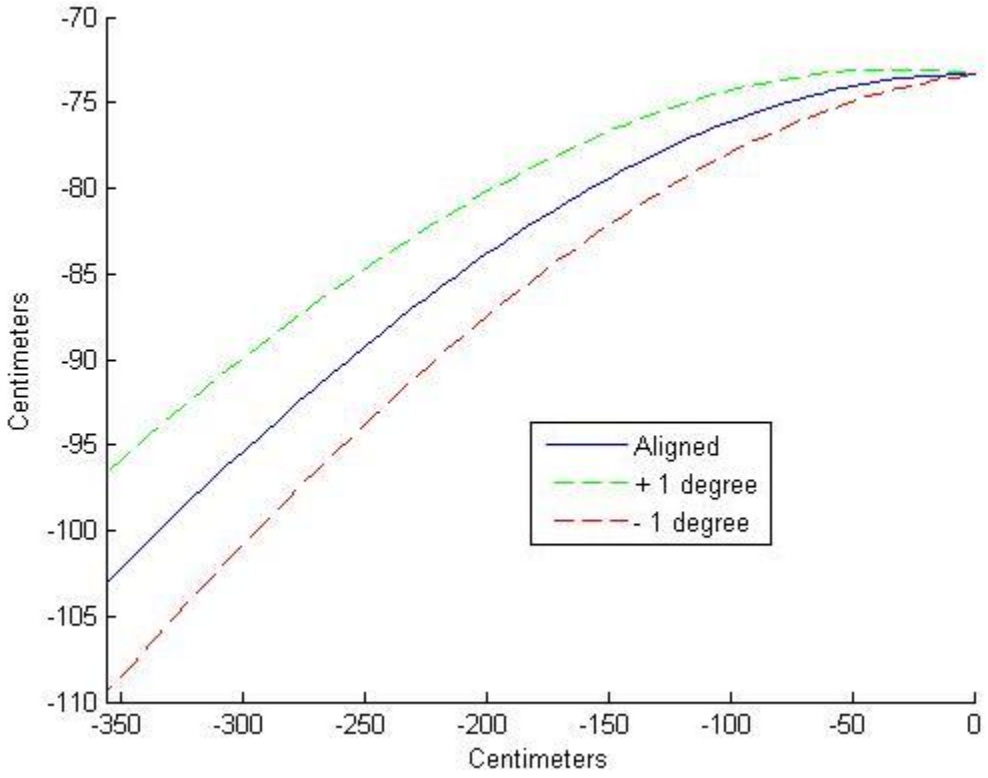

Figure 4.11: The variation in the hyperbolic correction from 1° misalignment of the pan/tilt device in roll for the bottom row of pushpins

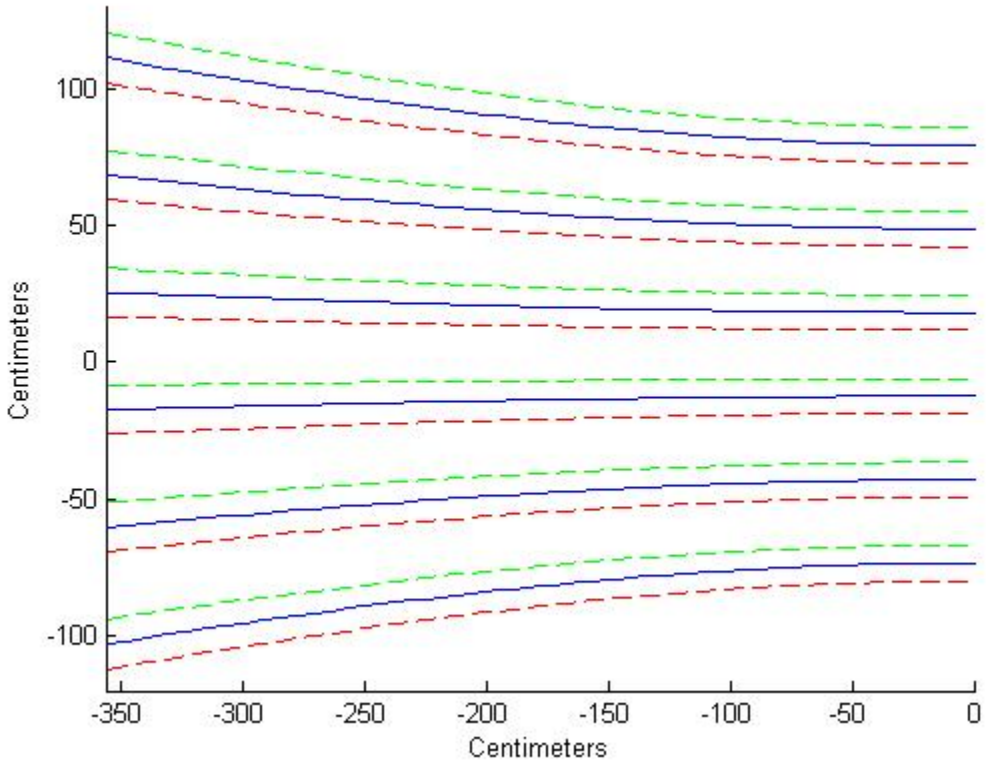

Figure 4.12: The variation in the hyperbolic correction from 1° misalignment of the pan/tilt device in pitch

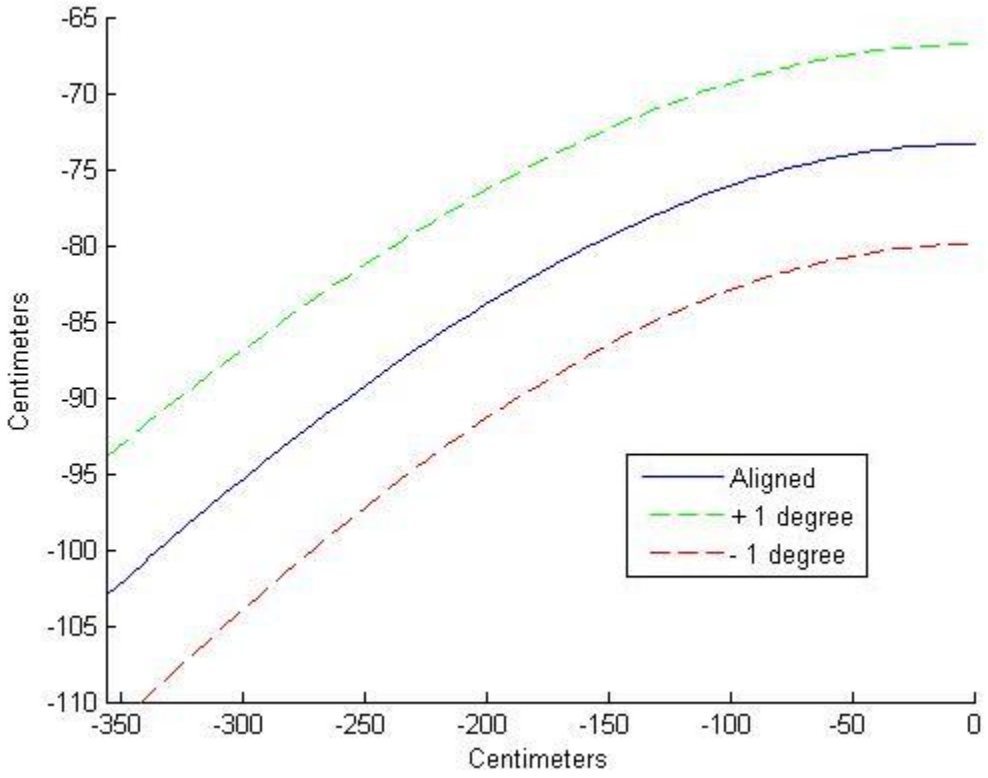

Figure 4.13: The variation in the hyperbolic correction from 1° misalignment of the pan/tilt device in pitch for the bottom row of pushpins

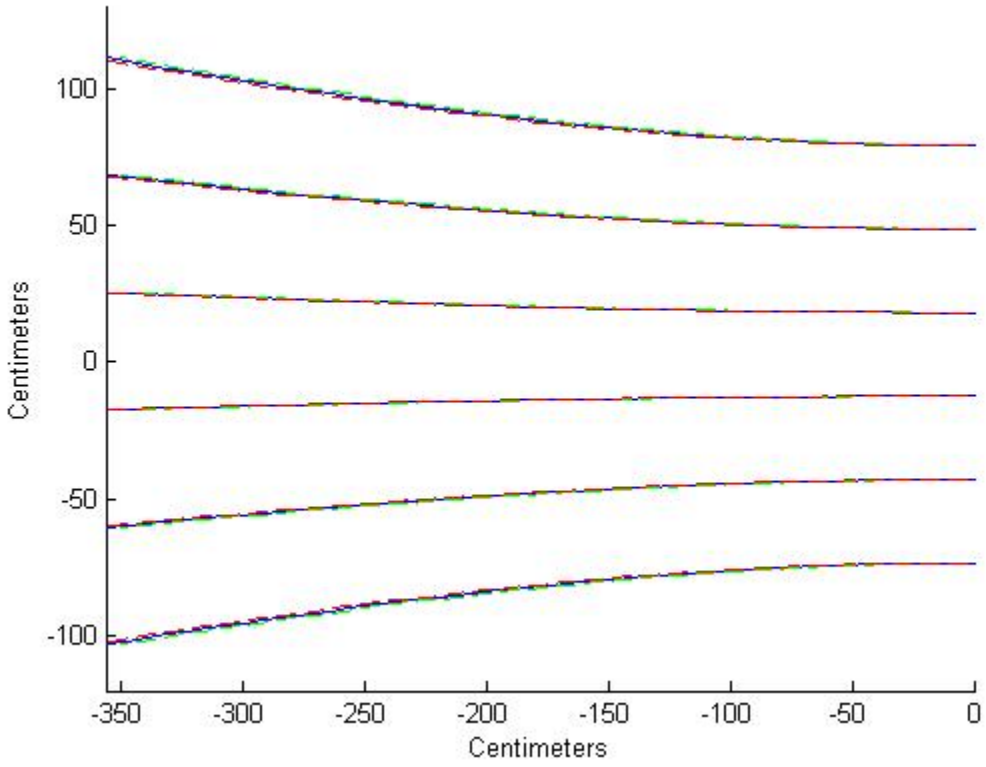

Figure 4.14: The variation in the hyperbolic correction from 1° misalignment of the pan/tilt device in yaw

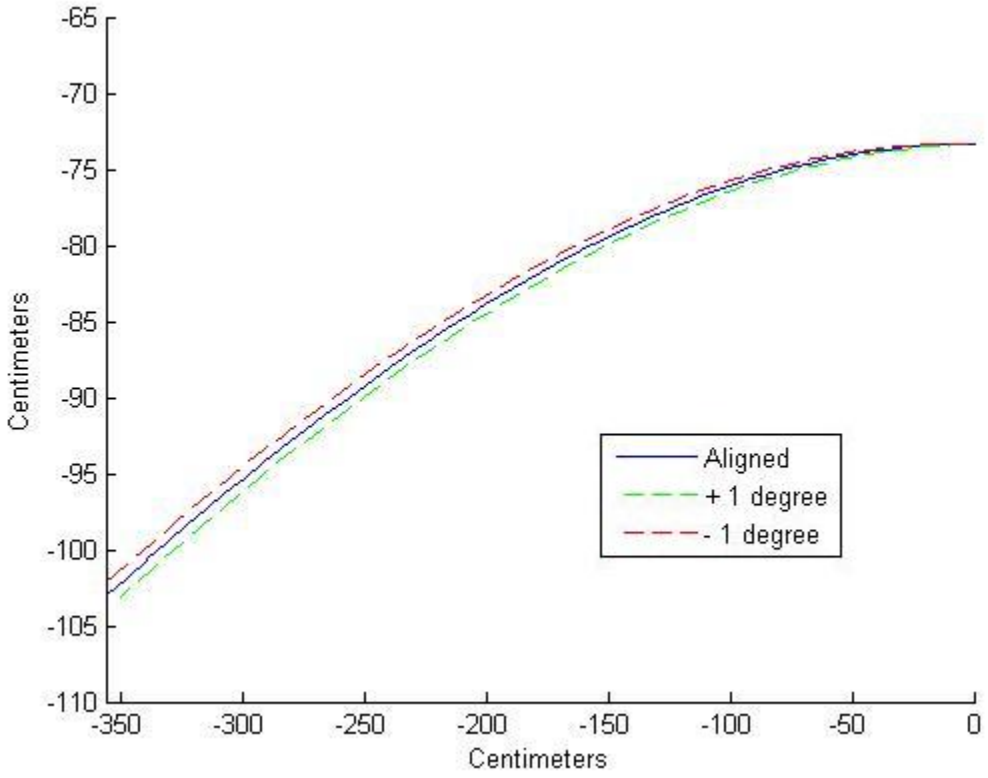

Figure 4.15: The variation in the hyperbolic correction from 1° misalignment of the pan/tilt device in yaw for the bottom row of pushpins

From these maps, misalignment in rotation has a much greater effect than misalignment in translation. Also, after closely comparing Figure 4.15 with Figure 4.3, it can be seen that deviation in yaw causes the type of errors seen in Figure 4.3. This makes is reasonable since yaw misalignment could not be detected by using a simple level like the pitch and roll. Calculations show that the error in the model test was caused by a an approximate 4° misalignment. Deviations in the translation cases can be easily determined by making careful measurements with a measuring tape. This is sufficient

evidence that the geometric model used was correct and also shows that great care must be taken in setting up the pan/tilt device and defining its location.

4.3 Static Targeting Accuracy Measurements

Once the geometric model used to control the system was verified, the static accuracy of the system was determined. At this point, the calibration lookup table was added to the program to correct for the misalignment of the system and optical distortion. The results of the test are shown in Figure 4.16.

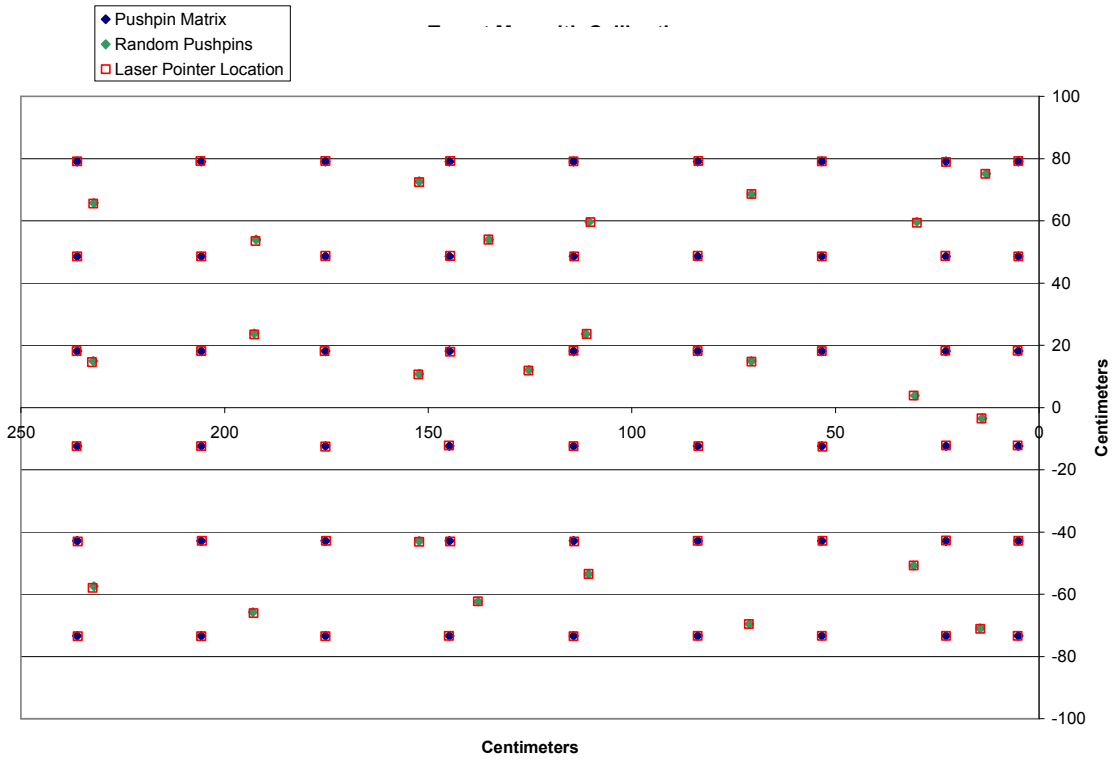

Figure 4.16: Results of static targeting with calibration

The system responded very well and accurately targeted all the pushpins. Table 4.1 gives the results of this test.

Table 4.1: Summary of data for the static targeting accuracy test

	Error/Pushpin (cm/pp)	Average Error (cm)	Standard Deviation (cm)	Max Error (cm)
Initial Pushpins	0.135	0.149	0.123	0.159
Random Pushpins	0.284	0.284	0.203	0.476
All Pushpins	0.181	0.194	0.166	0.476

The initial pushpins were targeted more accurately than the random pushpins. This was to be expected because the error for those locations was determined previously and used to construct the calibration table. The calibration for the random pushpins had to be interpolated from the other values. It should be noted that the resolution of the pan/tilt device was not always precise enough to land on the intended target. In these cases, the closest the laser pointer could get to the targeted location was approximately .159 cm away in either direction, i.e. horizontal and vertical. Using Equation 3-4, the average step distance of the laser pointer light across the image was calculated to be .143 cm. This is on the same order as the approximate value of .159 cm. This loss of resolution was distributed throughout the target plane.

For example, eight targets around the perimeter of the pushpin grid and one target in the middle were selected. The resolution of the pan/tilt device was determined at these locations. If the pan/tilt device was unable to land precisely on these targets, then the calculated error was corrected. This decreased the total error for these nine targets

from 1.43 cm to .953 cm. From this, it is obvious that the accuracy of the system could be increased with a hardware upgrade.

4.4 Static Targeting Processing Speed

Five time stamps were placed after the following five events occurred: acquisition of the pixel value and determination of the physical distances to the target from the origin, conversion of the distances into degrees, determination of the hyperbolic correction, determination the calibration correction, and commands sent to the pan/tilt device. Three trials were performed where the number of cycles completed was eighteen. Table 4.2 gives the average value of the elapsed time before each step occurred as well as the standard deviation of the total average.

Table 4.2: Tabulated processing times for the static targeting program

	Distance Determination	Degree Determination	Hyperbolic Correction	Calibration Correction	Command Sent
	Average	Average	Average	Average	Average
Run 1	$2.83*10^{-4}$ s	$4.00*10^{-4}$ s	$4.49*10^{-2}$ s	$4.50*10^{-2}$ s	$5.60*10^{-2}$ s
Run 2	$1.71*10^{-4}$ s	$2.62*10^{-4}$ s	$4.33*10^{-2}$ s	$4.40*10^{-2}$ s	$5.64*10^{-2}$ s
Run 3	$1.79*10^{-4}$ s	$2.72*10^{-4}$ s	$4.26*10^{-2}$ s	$4.26*10^{-2}$ s	$5.46*10^{-2}$ s
Average	$2.11*10^{-4}$ s	$3.11*10^{-4}$ s	$4.36*10^{-2}$ s	$4.39*10^{-2}$ s	$5.57*10^{-2}$ s
St Dev	$1.83*10^{-4}$ s	$3.97*10^{-4}$ s	$7.21*10^{-3}$ s	$7.23*10^{-3}$ s	$6.65*10^{-3}$ s

As the data show, the processing times are very fast. Since a lookup table approach was used for the calibration, the times could be even faster if the hyperbolic correction is simply accounted for in the calibration table rather than calculating it every time. The most time consuming process is the hyperbolic correction which uses the

curve fitting toolbox. Eliminating the need for this would increase the speed of the system.

5. TESTING OF DYNAMIC TARGETING CAPABILITY

5.1 Purpose and Objective

In addition to targeting stationary objects, the system's ability to target a moving object was tested as well. Since the accuracy of the system had already been determined, the goal of this part of the project was to determine the speed of the system as it tracked an object.

5.2 Experimental Setup

The setup for this experiment was the same as for the long range testing except for one difference. In the target plane, a pendulum was used instead of a series of pushpins. The 111.8 cm long pendulum contained an LED light and was attached to the wall. A piece of cardboard was cut into the shape of a circle so that a uniform bright image would be produced. This was intended to aid in the ease of calculating the centroid of the bright image. A picture of this is shown below in Figure 5.1.

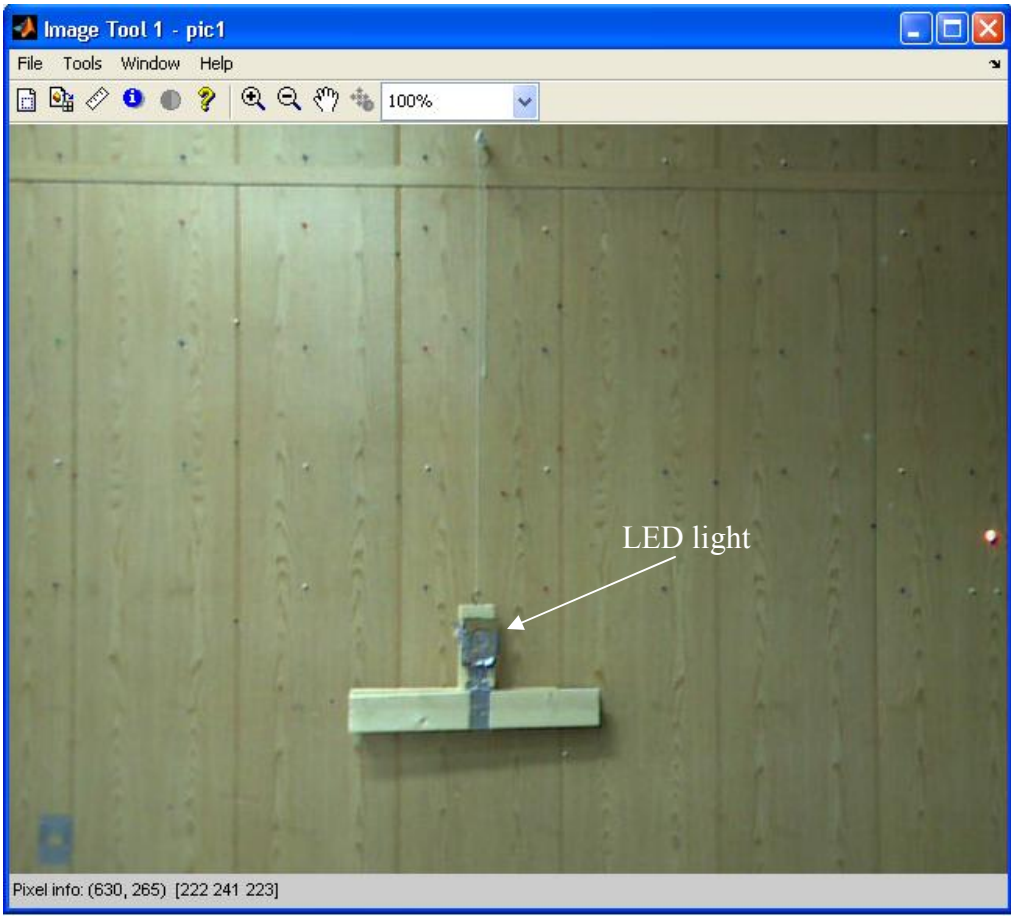

Figure 5.1: Pendulum used in the dynamic targeting experiment

5.3 Experimental Procedure

Initially, the pendulum was raised and held stationary along the wall. The tracking program was run and allowed to acquire the pendulum image location. Once the system had acquired the pendulum, the pendulum was allowed to swing freely. The targeting system was then required to re-acquire the target while it was in motion.

The speed of the system was determined by using the same method mentioned described previously. The performance of the laser pointer in tracking the LED light was

determined by looking at the frames of the video image. When the laser pointer light reached the calculated centroid position, the timestamp of that frame was compared to the timestamp of the acquisition of the image that contained the centroid.

5.4 Results and Discussion

Since the camera has variable frame rate settings, the rate at which Matlab processed images needed to be determined for each setting. It was not assumed that because the camera sent images to the computer at a certain rate, Matlab was able to process those images through the dynamic targeting program at the same rate. The following table (Table 5.1) gives the camera processing rates and the Matlab processing rates.

Table 5.1: Comparison of frame rates processing

Frame rate from camera	Rate at which Matlab processed frames
30 frames per second	~4.7 frames per second
25 frames per second	~4.7 frames per second
20 frames per second	~4.7 frames per second
15 frames per second	~4.7 frames per second
10 frames per second	~4.7 frames per second
5 frames per second	~2.5 frames per second

Since the speed of the algorithm depends on how fast Matlab can acquire an image and process it to determine a command for the pan/tilt device, tests performed at 30 fps through 10 fps should be the same. Data will be reported for only the 5 and 10 frames per second case and the 5 frames per second cases.

In order to understand the response of the various components of the system, the centroid's horizontal position was plotted versus the time it took for each process to complete for that position. Time stamps were placed after Matlab (1) acquired the image, (2) brought the image into the loop, (3) calculated the centroid, and (4) sent the ASCII command to the pan/tilt controller. The time for the mechanical response was determined by looking at the image frames and determining when the laser spot from the pan/tilt device arrived at the centroid that was calculated. The time stamp for that image was then used as the time of the mechanical response. This created a series of sinusoidal waves that were the same shape but out of phase with each other. The following figures show the response for the 10 frames per second case (Figure 5.2) and 5 frames per second case (Figure 5.3).

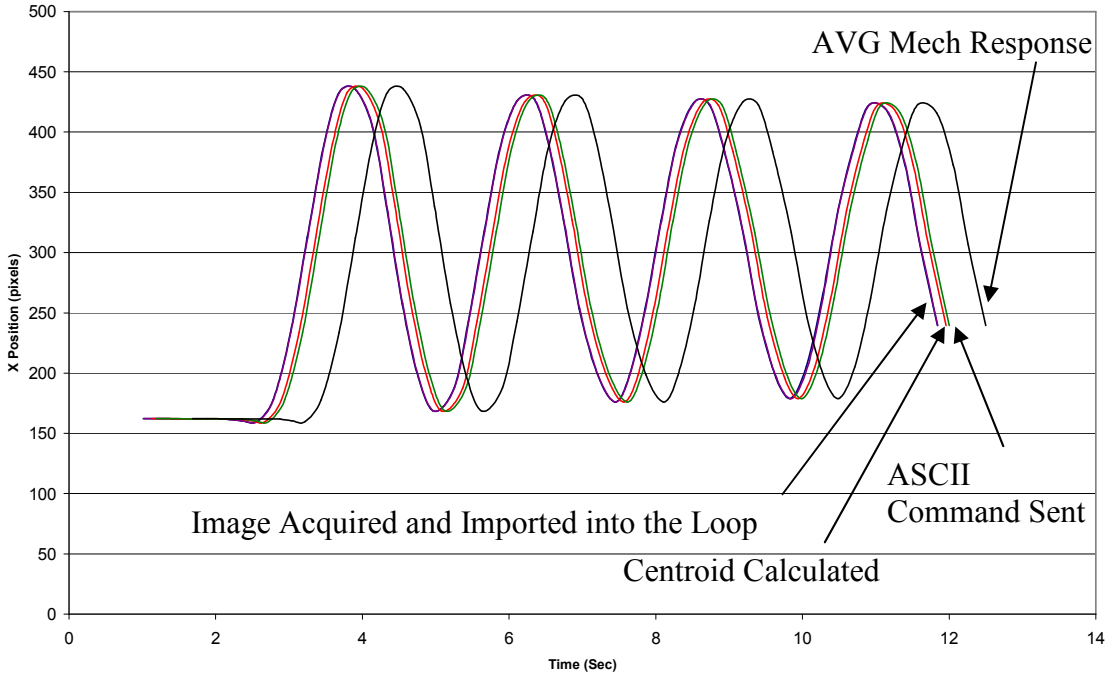

Figure 5.2: The response of the system at 10 frames per second

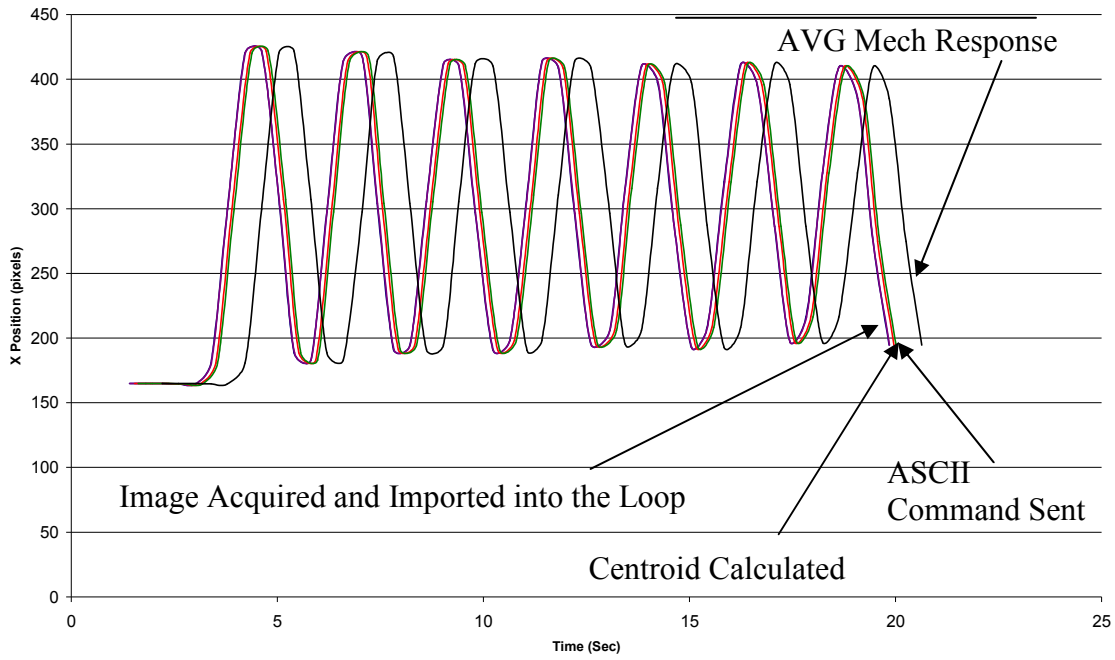

Figure 5.3: The response of the system at 5 frames per second

The difference between each curve was then determined and plotted versus number of iterations through the loop. These delays are shown below for the 10 frames per second case (Figure 5.4) and the 5 frames per second case (Figure 5.5).

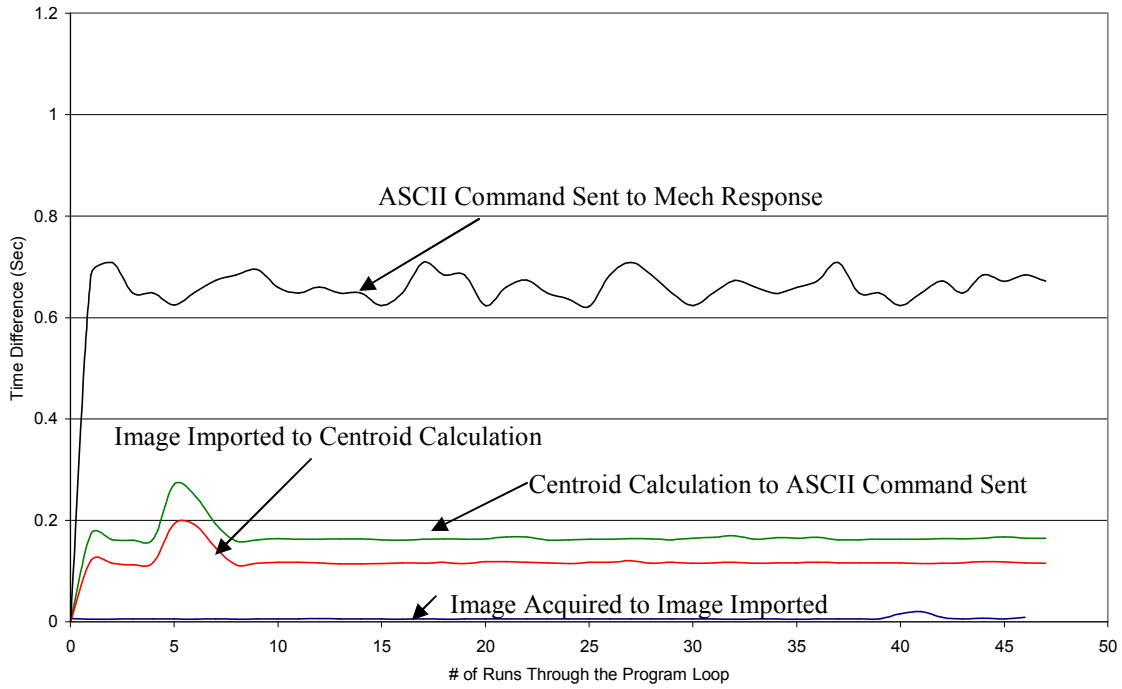

Figure 5.4: Delays associated with each process for 10 frames per second

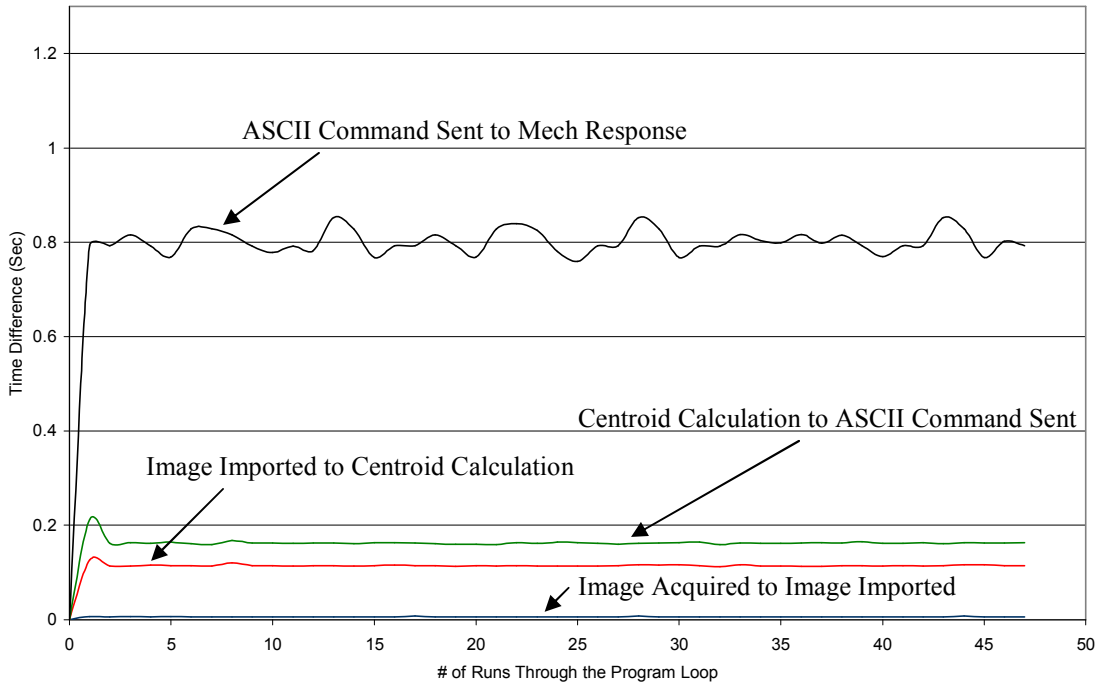

Figure 5.5: Delays associated with each process for 5 frames per second

In both cases, a significant delay occurred between the time that the ASCII command was sent and the time the pan/tilt/laser pointer system physically responded. In an attempt to better understand this phenomenon, the length of the pendulum was increased an additional 104.1 cm so that the string was 215.9 cm. The purpose of this was to decrease the speed of the pendulum. The reasoning behind this was that the pendulum was too fast for the system. By slowing it down, the system may have a chance to catch up which would help decrease the mechanical delay. However, experimental data showed that this change had no effect on the pan/tilt/laser pointer mechanical response and that the behaviors were identical for both the long and short pendulum.

In order to determine the origin of the time delays the program was rewritten line by line. After each new line was added, system speed was evaluated. By doing this, the exact point that the major delay was introduced was isolated. As long as the program was not waiting on an image, there were no delays in mechanical response.

The major time delays must lie with the data transfer from the USB camera to the host computer. USB is not the fastest available interface, and it can only respond to a host computer when the host computer communicates with it first [28]. An interface like FireWire, on the other hand, can send information without having to be called [28]. This could be part of the reason that the delay is occurring. Another possibility is that the USB camera may have an internal data buffer that is not documented in the literature [29]. This could be delaying the transfer of data and creating the mechanical delay. Also, although Matlab is a very robust program, it has limitations. As mentioned before, the images are coming into the environment at a much slower rate than they are capable of being sent. This, or quite possibly a combination of the three, could be causing the problem.

In order to determine how to alleviate this delay and increase performance, a series of tests were run where the resolution of the incoming image was changed and the method of image acquisition was changed (in Matlab there is more than one way to import an image into the dynamic targeting program loop). The two methods are using the functions *getsnapshot* or *peekdata*. This was done for the 10 frames per second case and the 5 frames per second case. Also, the average time needed to complete one loop was also recorded. The results of these tests are given below (Table 5.2). All of the times are in seconds.

Table 5.2: Results of the optimization trials for the dynamic targeting program

Condition	Avg Time Through the Loop		Mechanical Delay	
	5 fps	10 fps	5 fps	10 fps
GetSnapShot 640x480	0.1783	0.1850	0.698	0.498
GetSnapShot 320x240	0.1057	0.1159	0.638	0.419
PeekData 640x480	0.1923	0.1867	0.7	0.474
PeekData 320x240	0.1198	0.1130	0.635	0.448

The time through the loop is the same for all the cases. This indicates that it takes the same amount of time to import the image into the loop for all cases. The mechanical delay is reduced by using a smaller resolution; however, increasing the frame rate has the largest impact on the mechanical delay.

6. CONCLUSIONS

6.1 Camera Calibration

- The calibration process used was very simple and effective at a short range.
- The lens of the camera exhibited both pincushion and barrel distortion. The prevailing type of distortion was most easily seen around the perimeter.

6.2 Static Targeting

- The pan/tilt device must be properly aligned in order to produce accurate results.
- The geometric model used to model the system's environment was correct.
- The accuracy of the system could be increased by upgrading the current pan/tilt device to one with higher resolution.

6.3 Dynamic Targeting

- Tests at both 5 and 10 frames per second showed a significant delay as the system tracked the pendulum.

- The main delay of the system, i.e. the response of the pan/tilt device, resulted from the processing of images in Matlab.
- This delay was alleviated by both decreasing the resolution and increasing the frame rate. Increasing the frame rate reduced the system delay by approximately .2 seconds. Decreasing the resolution reduced the system delay by approximately .06 seconds.
- There is no noticeable difference in the performance of the system from using either *getsnapshot* or *peekdata* in the algorithm.

7. FUTURE WORK

Since proper alignment of the p/t device is essential to achieve accurate results, an investment should be made into constructing a stand that holds the device. This stand should have bubble levels and adjustment knobs so that the p/t device can be easily leveled. Also, a procedure for leveling the yaw must be developed since this cannot be done with a level. One way this could be done is by including a hinge so that the p/t device can be tilted 90° forward or backward. A level and adjustment knob could then be used to level the device. Once the device is leveled, it can be tilted back to its initial position. Another way would be to create a test target and aim at known locations. By comparing the results to the misalignment map, the degree of misalignment could be determined. Using an adjustment knob, this could then be corrected.

As previously stated, the current system uses only one camera which effectively limits its usefulness. A multi-camera system could stereographically determine distances to targets eliminating the requirement of knowing the distance to the target. This would allow for the system to target objects in a volume instead of on a plane.

If the system is to be used for tracking in its current setup, improvements to the image processing time must be made. This could be done by using a frame grabber to import images into the Matlab environment at rates closer to the camera's acquisition

rate, i.e. 30 frames per second. Also, a camera with a faster interface than USB such as FireWire could be used. Another would be using a different software platform specifically designed to process images.

However, the current setup is very limited in that it is designed to track a light in a dark environment. To make the system more applicable to a real world scenario, a different approach must be taken. In order to track a moving object in the light, such as a person, an approach such as tracking the color of piece of clothing, such as a shirt, should be used. Cameras with this capability are commercially available. For example, Carnegie Mellon's CMUcam has onboard computing ability to track color blobs.

Since the ultimate goal of this system is track or target a criminal and deliver a payload to stop any malignant actives, ballistic studies will have to be performed. A payload delivery system will have to be created and mounted onto the p/t device. Ballistic calculations will have to be made since the projectile will not travel in a straight line like a laser. Additional corrections will then have to be made to the geometric model to account for trajectory, wind resistance, etc. Also, a study will need to be done to see not only how the p/t device reacts from any recoil from the payload delivery system but also if the forces associated with payload delivery affects the system's performance.

REFERENCES

[1] Bureau Justice Statistics, "Four measures of serious violent crime," *U.S. Department of Justice*, Sept. 2006. [Online]. Available: http://www.ojp.usdoj.gov/bjs/glance/tables/4meastab.htm. [Accessed: May 2, 2008].

[2] Federal Bureau of Investigation, "Preliminary Semiannual Uniform Crime Report," *Federal Bureau of Investigation*, Jan. 2008. [Online]. Available: http://www.fbi.gov/ucr/prelim2007/table3.htm. [Accessed: May 2, 2008].

[3] Directed Perception "BLAST-Next Generation Intrusion Protection," *Directed Percpetion*, 2008. [Online]. Available: http://www.dpercpetion.com/pdf/SAIC-customer-story.pdf. [Accessed: May 2, 2008].

[4] B. Walker, *Optical Engineering Fundamentals*. Bellingham, WA: SPIE Optical Engineering Press, 1998, pp. 129-148.

[5] R. A. Serway, *Principles of Physics*. 2nd Edition, Fort Worth, TX: Saunders College Publishing, 1988, pp. 757-778.

[6] "Spherical Aberration," *wikipedia.com*, Apr. 21, 2003. [Online]. Available: http://en.wikipedia.org/wiki/Spherical_aberration. [Accessed May 2, 2008].

[7] A. Walther, *The Ray and Wave Theory of Lenses*. Cambridge, U.K.: Cambridge University Press, 1995, pp.51-68.

[8] R. Kingslake, *Lens Design Fundamentals*. New York, NY: Academic Press, 1978, pp 197.

[9] "Image Distortion," *wikipedia.com*, May 31, 2005. [Online]. Available: http://en.wikipedia.org/wiki/Image_distortion. [Accessed May 2, 2008].

[10] Z.Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, Nov. 2000.

[11] M. Xiaoqiao and H. Zhanyi, "A New Easy Camera Calibration Technique Based on Circular Points," *Pattern Recognition*, vol. 36, no. 5, pp 1155-1164, May 2003.

[12] K. Kanatani, *Geometric Computation for Machine Vision*. Oxford, U.K.: Clarendon Press, 1993.

[13] H. Zuech, *Understanding and Applying Machine Vision.* New York, NY: Marcel Dekker, 2000.

[14] M.Z. Abdullah, A.S. Fathinul-Syahir, and B.M.N. Mohd-Azemi, "Automated inspection system for colour and shape grading of starfruit (Averrhoa carambola L.) using machine vision sensor," *Transactions of the Institute of Measurement and Control*, vol. 27, no. 2, pp 65-87, 2005.

[15] I. Kunttu, L. Lepistö, J. Rauhamaa, and A. Visa, "Fourier-based object description in defect image retrieval," *Machine Vision and Applications*, vol. 17, pp. 211-218, June 2006.

[16] A. Higgins, S. Kouchy, and S. Mraz, "Machine vision gives security cameras an edge," *Machine Design*, vol. 75, no. 14, p. 14, July 2003.

[17] B. Bodenheimer, C. Rose, S Rosenthal, and J. Pella, "The Process of Motion Capture: Dealing with the Data," in *Eurographics Workshop on Animation and Simulation,* 1997.

[18] E.V. Biryukova, A. Roby-Brami, A.A. Frolov, and M. Mokhtari, "Kinematics of human arm reconstructed from spatial tracking system recordings," *Journal of Biomechanics*, vol. 33, pp. 985-995, 2000.

[19] C. Barron and I. A. Kakadiaris, "Estimating Anthropometry and Pose from a Single Image," in *IEEE Conference on Computer Vison and Pattern Recognition*, 2000, pp. 1669-1676.

[20] E. J. Alexander and T. P. Andriacchi, "Improving the analysis of human movement using markerless motion capture" presented at Summer Bioengineering Conference, Key Biscayne, FL, June 25-29 2003.

[21] N. Davison, "The Early History of "Non-Lethal" Weapons," Universtiy of Bradford, Bradford, U.K., Occ. Paper 1, Dec. 2006.

[22] J. Prendergast, "Beanbag shooting still unresolved," *The Cincinnati Enquirer*, Jan. 27, 2002. [Online]. Available: http://www.enquirer.com/editions/2002/01/27/loc_beanbag_shooting.html. [Accessed May 2, 2008].

[23] J. Dodge, "Military Beams over New Non-Lethal Weapon," *Design News,* vol. 63, no. 7, pp. 61-62, May 2008.

[24] M. Hall and E. Moreno, “Feds test weapon that can temporarily blind criminals,” *USA Today*, Aug. 8, 2008.

[25] J. Geralds, "Flashlight Weapon Makes Targets Throw Up," *Foxnews.com*, Aug. 27, 2007. [Online]. Available: http://www.foxnews.com/story/0,2933,292271,00.html. [Accessed May 2, 2008].

[26] “The future of crowd control,” *Economist*, vol. 373, no. 8404, p. 11, Dec. 2004.

[27] "Flight Dynamics," *wikipedia.com*, June 20, 2006. [Online]. Available: http://en.wikipedia.org/wiki/Flight_dynamics. [Accessed May 2, 2008].

[28] “FireWire – USB Comparison” *USB-WARE*, 2008. [Online]. Available: http://en.wikipedia.org/wiki/Flight_dynamics. [Accessed May 2, 2008].

[29] Joseph Cali (private communication), 2008.

APPENDICES

APPENDIX A

PAN/TILT DEVICE TECHNICAL SPECIFICATIONS

Table A.1: Specifications of Pan/Tilt Device

Payload	6 lbs
Max Unloaded Speed	300°/Second
Resolution	.0514°
Tilt Range	-47° to 31°
Pan Range	+/- 159°
Acceleration/Deceleration	Trapezoidal
P/T Unit Weight	3 lbs
P/T Unit Dimensions	3” wide x 5.13” high x 4.25” deep
Mounting	¼-20 screw holes
Software Controls	ASCII, Binary
Power	9 to 30 V DC capable of 17 W continuous

APPENDIX B

CODING

B.1 Hardware/Software Integration

In order for the p/t device and camera to be integrated together, both devices must first be brought into the Matlab environment. To establish communication with the p/t device, the Matlab function *serial* must be used. This function creates an object in the Matlab workspace that communicates with a device connected to a specified serial port. Once communication has been established, several properties of the object need to be specified. The manufacturer recommend settings, which were the values used, are as follows: a baud rate of 9600, no parity, 1 start bit, 8 data bits transmitted, and 1 stop bit. An example of this code is given below.

```
% Creates serial object, sets parameters, and opens object
s=serial('COM1');
set(s,'BaudRate',9600,'Parity','none','DataBits',8,'StopBits',1);
fopen(s);
```

Access to the Image Acquisition Toolbox is required to bring the live video from the camera into the Matlab environment. This is accomplished by creating a video object

with the desired resolution (as long as it is supported by the camera) using the *videoinput* function. After the video object is created, a figure and an image object need to be created. The image object contains the live video, and the figure contains the image object. The *preview* function is then used to display the video object inside the image object. An example of this code is given below.

```
% Creates video object
vid = videoinput('winvideo',1,'rgb24_640x480');

% Create a figure window. This turns off the default
% toolbar and menubar in the figure.
hFig = figure('Toolbar','none',...
'Menubar', 'none',...
'NumberTitle','Off',...
 'position',[350 300 640 480]);

% Create the image object in which you want to
% display the video preview data.
vidRes = get(vid, 'VideoResolution');
imWidth = vidRes(1);
imHeight = vidRes(2);
nBands = get(vid, 'NumberOfBands');
hImage = image( zeros(imHeight, imWidth, nBands));

% Specify the size of the axes that contains the image object
%  so that it displays the image at the right resolution and
%  centers it in the figure window.
figSize = get(hFig,'Position');
figWidth = figSize(3);
figHeight = figSize(4);
set(gca,'unit','pixels',...
'position',[ ((figWidth - imWidth)/2)...
((figHeight - imHeight)/2)...
imWidth imHeight ]);
```

```
% Displays the video object in the image object
preview(vid, hImage);
```

Once the hardware has been introduced into the software environment, the next step is to integrate the two together so that an input from the video image can be used to produce a response in the p/t device. This can be done in a number of ways. One way is to click on the video image. The depression of the mouse button causes Matlab to perform a task (in this case it would be to send a command to the p/t device). This can be done by using the figure property *windowbuttondownfcn* to call a function that would command the p/t device to move. Another approach would be to use the Matlab function *waitforbuttonpress*. This function causes Matlab to pause until a mouse button is depressed or a key is stroked. After a button press is registered, the program would resume operation in order to send a command to the p/t device.

Another way would be to send a command to the p/t every time the cursor moves to a new pixel inside the image. This is accomplished by using the figure property *windowbuttonmotionfcn*. When the cursor moves inside the image, a function can be called to control the p/t device.

A combination of the two can also be used. This occurs when the p/t device responds to a command when the mouse button is depressed. If the button is held in the depressed position while the cursor is moved from pixel to pixel, a command can be executed using the figure properties *windowbuttondownfcn* and *windowbuttonmotionfcn* in tandem. When the operator wishes to cease the function called by the

windowbuttonmotionfcn property, the figure property *windowbuttonupfcn* is used to delete the called function when the mouse button is released.

B.2 Static Targeting Program Code

```
%Closes any figures, clears any serial objects still open, deletes all variables, clears
%command window

Close all
f=instrfind;
if isempty(f)==0
   fclose(f);
   clear f;
else
   clear f;
end
clc
clear

%reset the image acquisition

imaqreset

% Create a video input object.

vid = videoinput('winvideo',1,'rgb24_640x480');
% obtain exclusive control of video object.

start(vid);


% Create a figure window. This turns off the default
% toolbar and menubar in the figure.

hFig = figure('Toolbar','none',...
   'Menubar', 'none',...
   'NumberTitle','Off',...
   'Name','TARGETING WINDOW','position',[350 300 640 480]);
```

```
%Changes the pointer to a crosshair while it is in the window.

set(gcf,'Pointer','crosshair');


% Create the image object in which you want to
% display the video preview data.

vidRes = get(vid, 'VideoResolution');
imWidth = vidRes(1);
imHeight = vidRes(2);
nBands = get(vid, 'NumberOfBands');
hImage = image( zeros(imHeight, imWidth, nBands));

% Specify the size of the axes that contains the image object
%  so that it displays the image at the right resolution and
%  centers it in the figure window.

figSize = get(hFig,'Position');
figWidth = figSize(3);
figHeight = figSize(4);
set(gca,'unit','pixels',...
   'position',[ ((figWidth - imWidth)/2)...
   ((figHeight - imHeight)/2)...
   imWidth imHeight ]);



% Displays the video object in the image object

preview(vid, hImage);


disp('To exit program, Hit any key while pointer is in figure window')

% w is perpendicular distance to target in feet (NOTE: ALL PHYSICAL
%   DISTANCES CALCULATED IN FEET)
%
% x is horizontal pixel value of initial laser pointer position, i.e.the
%   position of zero pan and zero tilt.  It is used as a reference point.
%
% p2d is the ratio of feet to pixels, i.e. the number of pixels that
%   represents a foot.
%
% key helps initate the while loop and is used to hold the response from
```

```
%   the waitforbuttonpress command (0 for mouse click, 1 for keystroke)
%
% adj_x_corr and adj_y_corr are the specific corrections for the
% established pushpin matrix

global pt
global adj_x_corr
global adj_y_corr

w=(142)/12;
x=634;
y=216;
p2d=.0129;

key=0;

% Creates serial object, sets parameters, and opens object
 s=serial('COM1');
 set(s,'BaudRate',9600,'Parity','none','DataBits',8,'StopBits',1);
 fopen(s);

while key == 0;

   key=waitforbuttonpress;

   % pt is the pixel value (horizontal, vertical)
   pt=get(gcf,'currentpoint');


   % deltax is the difference btw the input horizontal pixel and x
   deltax=-(pt(1)-x);

   % deltay is the difference btw the input veritcal pixel and the
   %   calculated reference
   deltay=(pt(2)-(y));

   % H is the physical distance of the vertical pixel input
   H=(p2d*deltay);
   % L is the physical distance of the horizontal pixel input
   L=(p2d*deltax);

   % thetax is the angle the p/t must pan to reach the target
   thetax=atand(L/w);
   % panpos is the angle converted into positions used by the p/t
   panpos=(thetax)/.0514;
```

```
% thetay is the angle the p/t must tilt to reach the target
thetay=atand(H/w);
% tiltpos is the angle converted into position used by the p/t
tiltpos=thetay/.0514;

% creates cone data for hyperbola determination
%
% h is the height of the imaginary cone created when the p/t device
%   tilts up or down.  It is created using an aribtray value for the
%   base that is further than w.
%
% ch is the profile curve.  Here, cylinder calcultes radii for each
%   vaule
h=(200/12)/tand(90-thetay);
ch=0:.01:(200/12);
[xx, zz, yy] = cylinder(ch, 200);


%FINDS DATA AT INTERSECTION OF CONE AND PLANE
% st is the indices of the data points that intersect the plane
st=find(zz>(w-.015) & zz<(w+.015));
% m and n are the dimensions of st
[m,n]=size(st);
% p is a place holder used to call the desired data
% xp and yp are the hyperbolic data points at a distance w
p=1;
while p<=m
   xp(p,1)=xx(st(p,1));
   yp(p,1)=h*yy(st(p,1));
   p=p+1;
end

% Fits a parabolic-shaped curve to the hyperbola
% c3 is the equation
[c3,c4]=fit(xp,yp,'poly5');


% geomH is the physical height of the hyperbola at the distance L
geomH=(c3(L));


% geomThetay is the angle the p/t must tilt to reach geomH
geomThetay=(atand(geomH/w));
% geomTP is geomThetay converted to positions used by p/t
geomTP=geomThetay/.0514;
```

```
   % diff is the absolute difference between the two posistions
   diff=abs(geomTP)-abs(tiltpos);

   % ftp is the final tilt position
   if pt(2)<=y
      ftp=tiltpos+diff;
   end
   if pt(2)>y
      ftp=tiltpos-diff;
   end


% tdeg is ftp rounded to the nearest integer
   tdeg=round(ftp);
% pdeg is the pan position rounded to the nearest integer
   pdeg=round(panpos);

  %total_correctoin is the specific program that uses weighted averages of
  %the established corrections in adj_x_corr and adj_y_corr to correct for
  %errors in between the established pushpins

   total_correction


   %corrh and corrv are the specific correction for the clicked push-pin
   %cpdeg and ctdeg are the corrected pan and tilt positions necessary
   % to aim at the desired push-pin
   corrh=adj_x_corr/12;
   corrv=adj_y_corr/12;
   cpdeg=pdeg-(atand(corrh/w))/.0514;
   ctdeg=tdeg+(atand(corrv/w))/.0514;


   % tstr and pstr converts ctdeg and pdeg to a string
   tstr=num2str(round(ctdeg));
   % tnstr and pnstr turns tstr and pstr into the ASCII numbers representing
   %   each character
   tnstr=double(tstr);
   % tr and pr add the values for "tp" (tilt position) and "pp" (pan position) to be included
   with the
   %   value sent to the p/t
   tr=[116 112 tnstr];
   % tc and pc are character arrays of the pan and tilt command
   tc=char(tr);
```

```
    pstr=num2str(round(cpdeg));
    pnstr=double(pstr);
    pr=[112 112 pnstr];
    pc=char(pr);




    % sends the pan and tilt command through s to the p/t device


    fprintf(s,tc);
    fprintf(s,pc);




    %Sends the p/t to the initial position if a keystroke is given
    if key==1
       chara=get(gcf,'CurrentCharacter');
       ptsetup
       ipstr=num2str(0);
       ipnstr=double(ipstr);
       ipr=[112 112 ipnstr];
       ipc=char(ipr);
       itstr=num2str(0);
       itnstr=double(itstr);
       itr=[116 112 itnstr];
       itc=char(itr);
       fprintf(s,ipc)
       fprintf(s,itc)
       delete(s)
       clear s
       stoppreview(vid)
       close(gcf)

    end

end
```

B.3 Dynamic Targeting Program Code

```
%Closes any figures, clears any serial objects still open, deletes all variables, clears
%command window

Close all
f=instrfind;
if isempty(f)==0
   fclose(f);
   clear f;
else
   clear f;
end
clc
clear

%reset the image acquisition

imaqreset


% Create a video input object.

vid = videoinput('winvideo',1,'rgb24_640x480');

% obtain exclusive control of video object.

start(vid);


% Create a figure window. This turns off the default
% toolbar and menubar in the figure.

hFig = figure('Toolbar','none',...
   'Menubar', 'none',...
   'NumberTitle','Off',...
   'Name','TARGETING WINDOW','position',[350 300 640 480]);


%Changes the pointer to a crosshair while it is in the window.

set(gcf,'Pointer','crosshair');
```

```
% Create the image object in which you want to
% display the video preview data.

vidRes = get(vid, 'VideoResolution');
imWidth = vidRes(1);
imHeight = vidRes(2);
nBands = get(vid, 'NumberOfBands');
hImage = image( zeros(imHeight, imWidth, nBands));

% Specify the size of the axes that contains the image object
%  so that it displays the image at the right resolution and
%  centers it in the figure window.

figSize = get(hFig,'Position');
figWidth = figSize(3);
figHeight = figSize(4);
set(gca,'unit','pixels',...
   'position',[ ((figWidth - imWidth)/2)...
   ((figHeight - imHeight)/2)...
   imWidth imHeight ]);


% Displays the video object in the image object
preview(vid, hImage);

%centroid is the pixel value of the centroid
   %centy is used to correct the fact that the pixel tool uses the bottom pixel as the
   % first pixel while the program uses the top pixel as the first pixel
   %adj_x_corr and adj_y_corr are the specific corrections for the
   % established pushpin matrix


   global centroid
   global centy
   global adj_x_corr
   global adj_y_corr

   % w is perpendicular distance to target in feet (NOTE: ALL PHYSICAL
   %   DISTANCES CALCULATED IN FEET)
   %
   % x is horizontal pixel value of initial laser pointer position, i.e.the
   %   position of zero pan and zero tilt.  It is used as a reference point.
   %
   % p2d is the ratio of feet to pixels, i.e. the number of pixels that
```

```
%   represents a foot.
%
% key helps initate the while loop and is used to hold the response from
%   the waitforbuttonpress command (0 for mouse click, 1 for keystroke)
%




w=(142-0)/12;
x=634;
y=216;
p2d=.0129;
key=0;
% Creates serial object, sets parameters, and opens object
s=serial('COM1');
set(s,'BaudRate',9600,'Parity','none','DataBits',8,'StopBits',1);
fopen(s);




key=waitforbuttonpress;

%sets the number of frames aqcuired per triggering of program, the delay of
% of how many frames are aqcuired before the program starts using frames,
% and how many times the trigger is repeated

vid.FramesPerTrigger = 200;
vid.TriggerFrameDelay = 0;
vid.TriggerRepeat = 0;

%starts and triggers video object
start(vid);
trigger(vid);

while key == 0 & islogging(vid)==1;

    % pic is an image from the video stream
    pic=getsnapshot(vid);

    %BW is pic converted to black and white
    %BW2 filters out light form laser pointer
```

```
%l lables BW2
%rp sets the region properties while centroid is the pixel vaule of the
%center of the light being tracked
BW=im2bw(pic,.5);
BW2 = bwareaopen(BW,100);
l=bwlabel(BW2);
rp = regionprops(L, 'centroid');
centroid = cat(1, rp.Centroid);



% deltax is the difference btw the input horizontal pixel and x
% deltay is the difference btw the input veritcal pixel and the
%   calculated reference
% H is the physical distance of the vertical pixel input
% L is the physical distance of the horizontal pixel input


deltax=-(round(centroid(1))-x);
centy=480-round(centroid(2));
deltay=(centy-(y));
H=(p2d*deltay);
L=(p2d*deltax);


% thetax is the angle the p/t must pan to reach the target

% panpos is the angle converted into positions used by the p/t
% thetay is the angle the p/t must tilt to reach the target
% tiltpos is the angle converted into position used by the p/t

thetax=atand(L/w);
panpos=(thetax)/.0514;
thetay=atand(H/w);
tiltpos=thetay/.0514;




% creates cone data for hyperbola determination
%
% h is the height of the imaginary cone created when the p/t device
```

```
%   tilts up or down.  It is created using an aribtray value for the
%   base that is further than w.
%
% ch is the profile curve.  Here, cylinder calcultes radii for each
%   vaule
h=(200/12)/tand(90-thetay);
ch=[0:.01:(200/12)];
[xx, zz, yy] = cylinder([ch], 200);


%FINDS DATA AT INTERSECTION OF CONE AND PLANE
% st is the indices of the data points that intersect the plane

% m and n are the dimensions of st
% p is a place holder used to call the desired data
% xp and yp are the hyperbolic data points at a distance w

st=find(zz>(w-.015) & zz<(w+.015));
[m,n]=size(st);
p=1;
while p<=m
   xp(p,1)=xx(st(p,1));
   yp(p,1)=h*yy(st(p,1));
   p=p+1;
end

% Fits a parabolic-shaped curve to the hyperbola
% c3 is the equation
[c3,c4]=fit(xp,yp,'poly5');



% geomH is the physical height of the hyperbola at the distance L
% geomThetay is the angle the p/t must tilt to reach geomH
% geomTP is geomThetay converted to positions used by p/t
% diff is the absolute difference between the two posistions
% ftp is the final tilt position

geomH=(c3(L));
geomThetay=(atand(geomH/w));
geomTP=geomThetay/.0514;
diff=abs(geomTP)-abs(tiltpos);
if centy<=y
   ftp=tiltpos+diff;
end
```

```
if centy>y
   ftp=tiltpos-diff;
end



% tdeg is ftp rounded to the nearest integer

% pdeg is the pan position rounded to the nearest integer


tdeg=round(ftp);
pdeg=round(panpos);

%total_correctoin_track is the specific program that uses weighted averages of
%the established corrections in adj_x_corr and adj_y_corr to correct for
%errors in between the established pushpins

total_correction_track

%corrh and corrv are the specific correction for the clicked push-pin
%cpdeg and ctdeg are the corrected pan and tilt positions necessary
% to aim at the desired push-pin

corrh=adj_x_corr/12;
corrv=adj_y_corr/12;
cpdeg=pdeg-(atand(corrh/w))/.0514;
ctdeg=tdeg+(atand(corrv/w))/.0514;



% tstr and pstr converts ctdeg and pdeg to a string

% tnstr and pnstr turns tstr and pstr into the ASCII numbers representing
%   each character

% tr and pr add the values for "tp" (tilt position) and "pp" (pan position) to be
%included with the
%   value sent to the p/t

% tc and pc are character arrays of the pan and tilt command
tstr=num2str(round(ctdeg));
tnstr=double(tstr);
tr=[116 112 tnstr];
tc=char(tr);
```

```
    pstr=num2str(round(cpdeg));
    pnstr=double(pstr);
    pr=[112 112 pnstr];
    pc=char(pr);

    % sends the pan and tilt command through s to the p/t device

    fprintf(s,tc);
    fprintf(s,pc);


end
```