Project Number: ME-CDO-1301/1302

WPI

Motor Eyes

Mechanical Platform for a Binocular Robotic Vision System

A Major Qualifying Project Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science

Submitted by:

William Kiely
Wut Yee Oo
Olive Rappoli
Brian Strobel
Raphael Walcott

Advised by:

Professor Cagdas D. Onal
Professor Holly K. Ault

April 30, 2014

Abstract

Stereoscopic vision systems, which introduce depth to an image by overlaying two separate perspectives of the image, require high computational power to perform image processing. The required processing power can be reduced by synchronizing eye movements to direct their respective lines of sight to the same point, aligning the images. To reduce synchronization error, eye movements can be coupled mechanically. To investigate this potential, the project team developed, analyzed, constructed, and tested a mechanical platform for a binocular robotic vision system. The design used linear motion and slider linkages to couple the eye movements, with each degree of freedom being actuated by a stepper motor. The motor positions were recorded by encoders and controlled by an Arduino Uno and Adafruit Motor Shield. A prototype mechanical platform was constructed and tested for eye velocity and positional accuracy. The maximum angular eye velocity achieved was 230.8 degrees per second, approximately 46% of the maximum angular velocity of the human eye. The prototype was sensitive to misalignment in the mechanical components, occasionally leading to large error in the focus point position. However, testing confirmed that the prototype was capable of focusing on points that were reasonably close to the predicted locations, particularly at zero pan and close convergence depths. Building the prototype also confirmed that the mechanism was capable of pointing two eye components at an individual point of interest through mechanically coupled pan, coupled tilt and coupled vergence movements.

Acknowledgements

We would like to first thank our advisors, Professor Cagdas Onal and Professor Holly Ault, for their guidance throughout the duration of this project and making the MQP possible. We would like to thank Professor Onal for envisioning the concept behind the project and providing us with support and technical experience throughout the design process. We would like to thank Professor Ault for advising us on design considerations and mechanism analysis, and for her keen attention to detail.

We are also indebted to David Ephraim, our contact in the Washburn Machine Shop at WPI, who assisted us with our machining needs very well and in a timely manner. We want to thank Joe St. Germain, WPI's Robotics Lab technician, for allowing us access to mechanical components used for prototyping and testing; and the ECE Shop for addressing our needs for soldering and wiring.

Finally, we would like to thank Barbara Furhman for her help with purchasing parts, and Statia Canning for answering our logistical questions and her help with preparations for Project Presentation Day.

Table of Contents

List of Figures

List of Tables

Authorship

The design, analysis, construction and testing of the platform, as well as the majority of the final report, were contributed to equally by all five team members: William Kiely, Wut Yee Oo, Olive Rappoli, Brian Strobel and Raphael Walcott.

Wut Yee Oo developed the control system for the platform.

Wut Yee Oo and Raphael Walcott implemented the control system in the mechanical platform prototype and wrote “Chapter 8 Control System Architecture Design” and “Section 9.4 Implemented Control System Architecture.”

1 Introduction

Many stereoscopic vision systems in existence today use image processing methods in which an entire scene is analyzed at once. Such systems utilize the input images from two cameras that are fixed relative to one another. Each whole image is used to complete a 3D reconstruction of the environment, a method in which common corresponding features are identified in the two images to calculate the depth of the features in the scene. This image processing technique to determine the depths of objects in a scene requires a relatively simple mechanical platform, typically with the cameras fixed in place, but has high computational cost. Some robotic vision systems are capable of seeing everything that surrounds the system platform at the same time. This is useful from a surveillance standpoint because it maximizes the robot's range of vision; however, the extensive image processing may provide the system with more information about its surroundings than is necessary and may demand an unreasonably high amount of computing power (Murray, 1992).

The human vision system enables selective observation of certain parts of a scene. By orienting our eyes to focus on points of interest we can determine the relative depths of objects and see them in more detail. Objects in the center of our field of vision are focused while objects in the periphery are not in focus. However, we are capable of detecting motion in our peripheral vision and can shift our gaze to focus on new targets. The kinematics of the ocular muscles in each eye allows the eyes to move with agility to track objects smoothly as well as to move rapidly. This is called saccadic motion (Orban de Xivry, 2007). Much research has been conducted to develop robotic systems that implement a vision system that mimics the effectiveness of human vision (Murray, 1992). Such systems place more dependence on the mechanical platform of the system, and can be implemented to reduce the computational costs of extensive image processing. Many research projects have worked on modeling several different aspects of the functions of human vision, from directly mimicking the intricate kinematics of eye movements (Schultz, 2012) to developing an overall system capable of carrying out the selective-attention vision process that humans implement. These systems have been used to study the mechanics of the eye as well as applications in social robotics. Social robots are robots designed to interact with humans in a personal and intellectual manner. Our eye movements play an important role in the way we communicate with other humans, and technology that mimics human vision lends itself to creating humanoid robots with artificial intelligence that can interact with humans (Breazeal, 2001).

Social robots have applications in personalizing and enhancing interactions with computerized interfaces, providing personal companionship and entertainment, and can also be used for therapy for children with autism. These applications rely on the robot achieving an emotional connection with the human, such that there is a mutual understanding of the facial expressions and other social indicators to enhance communication. A social robot should exhibit some of the same social abilities as a human in order for a natural connection to be made during human-robot interaction (Breazeal, 2001). The design of the robotic system must therefore be subject to social constraints, which influence the technological design and function of its components. Considerations include the range of motion and speed of eye components to achieve eye movements similar to that of humans. Additionally, the robotic eyes must be able to move in a synchronized manner and converge on single points of interest like human eyes can.

Existing robotic vision systems that are meant to realistically mimic these functions generally are not able to simultaneously converge both robotic eyes on single points of interest with the same accuracy as humans. Most existing robotic vision systems also fail to track objects in motion with the same high precision as human eyes. Consequently, the objective of this project is to develop a mechanical platform for a robotic vision system that can orient the robot's eyes to aim at points in a

scene with a high degree of accuracy similar that of human eyes. Additionally, this mechanical platform should be able to control the motion of the eye components to more accurately mimic the saccadic motion and tracking abilities of human eyes. A particular goal is to mechanically couple the movements of the two eyes for each type of eye movement (pan, vergence and tilt) so that eye synchronization can be achieved without complex programming. The long-term goal is for this mechanical platform to be used for future projects that can further develop the programming aspects of the system. We place our focus on developing a platform for a social robot application because of the technical challenges of creating a system that exhibits human-like behavior, and achieves these using primarily mechanical methods.

2 Background

This chapter contains background research relevant to our project. It provides information on: vision in humans and robots, the mechanisms behind the vision systems, and the functions of the vision systems. These topics are necessary for understanding how the human vision system works, and how it can be mimicked and used to formulate design specifications for a robotic vision system. Also discussed are technology and applications that have already been developed which are used as inspiration for this project's platform design.

2.1 Human Vision

The human vision system is mechanically complicated and serves many important functions. The processes involved in creating the images we as humans see are complex and require two input images, one from each eye. How the eyes and their movements are perceived by other humans is another important factor in non-verbal communication with significant social implications.

2.1.1 Functions and Purposes of Human Eyes

Human eyes are specialized and highly developed organs that serve many important functions in communication and perception of our surroundings. In particular, the image-processing and kinematic capabilities of our eyes are fundamental to several sensory and social functions that humans rely on in daily interactions with each other and the environment.

Sensory Functions

The most basic purpose of our eyes is to provide us with vision, a sensory input that humans rely heavily on to gain an understanding of our environment. Having two eyes enables stereovision, allowing us to clearly focus on, track, and determine the depth of objects of interest. We are able to distinguish between colors and other features to recognize objects and identify faces. Additionally, we are able to see to some extent through our peripheral vision. While images on the outskirts of our vision are not in focus (because they may not yet be of interest to the human), we are capable of detecting motion in our peripheral vision. These sensory capabilities of human eyes have developed for survival purposes. We are able to see an object in sufficient detail when it is focused in the center of our range of vision, and depth perception allows us to determine the location of objects or obstacles, contributing to navigation and orientation. However, motion detection in our peripheral vision alerts us to an object that may require attention, a survival response to react to threatening movement. These visual capabilities are evident in our daily lives – for example, while driving, we can focus on the road ahead and read street signs, but can also shift our gaze to another car, bicycle or pedestrian when their movement is detected in our peripheral vision (Montgomery, 2013).

Human eye motion is characterized by two types of movement: smooth pursuit and saccadic. Smooth pursuit is exhibited when tracking an object moving at a relatively low velocity. Saccadic movements are rapid eye movements, exhibited when a human is looking around, continuously shifting and adjusting focus to various objects in a scene (Orban de Xivry, 2007). Humans use these movements to observe their environment, but also to communicate with fellow humans, as discussed in the following section.

Social Functions

Different types of eye movements are indicative of what a person has decided to respond to (i.e., has determined to be relevant and deserving of attention) in a particular scene. For example, a steady gaze and smooth-pursuit eye movement indicates concentrated attention, whereas any fast saccadic motion used to glance at something indicates a brief shift of interest (Breazeal, 2001). Humans exhibit this behavior with their eye movement and are also able to observe and interpret eye movements as social cues in other humans. During interaction, humans tend to make eye contact to show that they are mutually directing their attention towards each other. Steady gazing, or staring, is interpreted differently from ordinary eye contact. Humans are able to understand the implications of someone's interest by their eye movements as well – for example, if a person sees another person's eyes exhibiting smooth pursuit, that person understands that a moving object is being tracked and can follow the person's gaze to find the object of interest. Additionally, eye movements coupled with head, eyelid and eyebrow movements contribute to producing facial expressions that humans use to gauge emotion (Breazeal, 2001).

2.1.2 Eye Movement Mechanism

The human eye is controlled by six extra-ocular muscles: the lateral rectus, medial rectus, superior rectus, inferior oblique, inferior rectus, and superior oblique. The muscles for each eye control the eye independently from the other. The lateral and medial recti can only move the eye in abduction, or horizontally about the vertical axis. Together, the superior rectus and inferior oblique cause the eye to elevate, moving up and down about a horizontal axis. Also, the superior oblique and the inferior oblique twist the eye, causing torsion about the direction of gaze of the eye (Quaia, 2001).

2.1.3 Image Processing

The human vision system can most simply be approximated by two elements: the eye, functioning as a camera, and the brain, functioning as a processor. Inside the eye, visual light is passed through the retina and then received by photoreceptor cells in the fovea, along the back of the eye (Thorpe, 1991). There are around 100 million of these photoreceptors in each eye, and are categorized as either a rod or a cone. Rods detect motion more than specific details of images, and are located around the edges of the eye, whereas cones are located more towards the center, and are used to distinguish color very precisely. The information gather by the rods and cones are sent back to the brain, via the optical nerve coming out of the back of the eye.

The two optic nerves meet near the back of the brain. At this point, the image inputs split based on the right and left halves of each image. The combined right halves of each image are sent to the left hemisphere of the brain for processing, and similarly the left halves of the images are sent to the right hemisphere. Once they are ready for processing, it takes between 0.1 and 0.15 seconds of processing an image for the brain to accurately determine what it is looking at (Thorpe, 1991).

2.2 Robot Vision

This section addresses robot vision systems. The functions and purposes of robotic vision for social robots are discussed, as well as the actuation and image processing used in robotic vision systems. Existing applications of robot vision are also discussed.

2.2.1 Robotic Vision for Social Robots

The field of social robotics relies heavily on robotic vision systems as a means of sensory input and output for a social robot interacting with a human and its surroundings. In the areas of artificial intelligence and humanoid robotics, the vision system of a robot plays an important role in controlling how the robot behaves – the vision system provides the robot with information about its surroundings, to which the robot must react appropriately and in a realistic manner. Much research has been done in developing personal robots that are capable of communicating with humans and responding to their environment using the same visual cues that humans use. This includes determining the location and depth of objects, detecting motion and tracking objects, and identifying eyes on a human face in order to make eye contact during human-robot interaction. Additionally, some social robots have been designed to learn. For example, some can be taught to identify objects and recognize human faces to distinguish between different people. If a social robot can be designed to share some of these social capabilities with a human, the human and robot will be better able to interpret each other's gestures as social cues, and the human will be able to understand and connect with the robot on a more emotional level (Breazeal, 2001).

The advancement of the field of social robotics has also helped to develop the interface between technological and social research. Linking the two fields of research by using sensor input with mechanics to mimic human behavior in a robot requires an understanding of human perception and communicative behavior. Developing computer vision systems to mimic the functions of human vision system (not only image processing but also the kinematics of human eye movement) has contributed to innovation in the programming and mechanical aspects of computer vision. For example, researchers at the Georgia Institute of Technology have developed a camera positioner that implements piezoelectric cellular actuators that are modeled closely after the muscles in human eyes, instead of more commonly used servo motors (Schultz, 2012).

In addition to applications in robotic research, personal robots are being developed for many purposes including entertainment as well as assistive technology, such as companionship and personalized assistance with daily tasks ("Personal Robots" 2013). Social robots may also be used in therapy treatments for children with autism. Children with autism often have difficulty using and interpreting facial expressions. Studies have shown that sometimes children with autism are more responsive to learning social cues by interacting with a social robot (Alexander, 2011). Advanced vision systems have been developed to make such applications possible, by providing the robots with the ability to not only interact with humans intelligently, but also to appear humanistic and allow the human to become emotionally connected with the robot. Social robots have the potential to someday become an engaging presence in our daily lives, capable of assisting with various tasks in the home or workplace while providing realistic social interaction and companionship ("Personal Robots" 2013).

2.2.2 Eye Movement Mechanisms

Robotic vision platforms can have a range of possible configurations and can use various kinds of actuators to control the motion of cameras or other sensors in a way that mimics the motion of human eyes. The following sections discuss some kinematic configurations and actuator types that have been implemented by previous research groups.

Mechanical Camera Platform Configurations

There are many different ways that a mechanical platform can be designed to move and orient cameras. Some of these designs give the cameras many degrees of freedom, while others give them only

a few degrees of freedom. What they have in common is how they control the motion of a pair of cameras to mimic the motion of human eyes. Two of the possible configurations for mechanical camera platforms, the common elevation configuration and the independent gun-turret configuration are described below.

Common Elevation Model:

One configuration of a mechanical platform that can move two cameras like human eyes is called the common elevation model. The common elevation model gives the two cameras a total of four degrees of freedom, as shown in Figure 1 (Murray, 1992). The first degree of freedom comes from the fact that the two cameras can rotate together on the horizontal plane, similar to turning one's neck to point the head in different directions horizontally. The second degree of freedom is that both eyes can tilt up and down together, as in nodding one's head. The third and fourth degrees of freedom come from the ability of each camera or eye to pan left and right independently. This ability to pan each camera independently allows the cameras to converge on points at different depths.

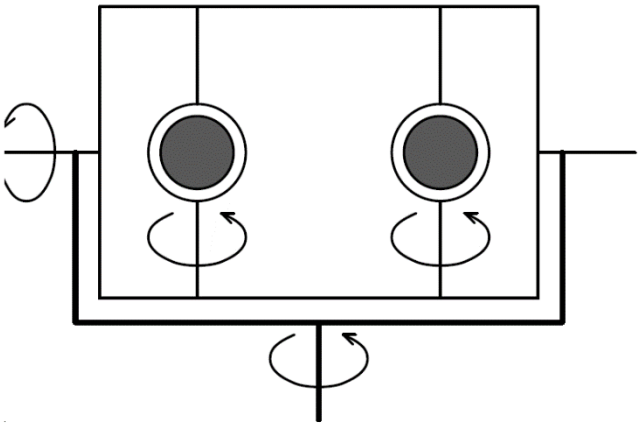


Figure 1: Common Elevation Model for a Sensor Platform (Murray, 1992).

The common elevation model has the advantage of a large range of vision, with both cameras being able to converge at any point in that range of vision. Additionally, it is a simple configuration that has only four degrees of freedom.

An interesting feature of the common elevation model is the fact that each camera cannot tilt up and down independently. While this may seem like a disadvantage at first glance, it should be noted that human eyes cannot tilt up and down independently either. Since the ability to tilt the two cameras or human eyes independently is not needed to make both cameras or eyes focus on a single point simultaneously, this constricted motion is most likely not a disadvantage of the common elevation model. In fact, this characteristic may actually be an advantage since both cameras will always converge on a single point, assuming they are angled towards one another.

Independent Gun-Turret Model:

The independent gun-turret model is similar to the common elevation model, except that it gives the two cameras one additional degree of freedom for five degrees of freedom in total (Murray, 1992). Like the common elevation model, the independent gun-turret model allows both cameras to rotate together on a horizontal plane, as a neck rotates the head to face left or right. Also like the

common elevation model, each camera can pan independently allowing for convergence. However, the independent gun-turret model enables the two cameras to tilt up and down independently, as shown in Figure 2.

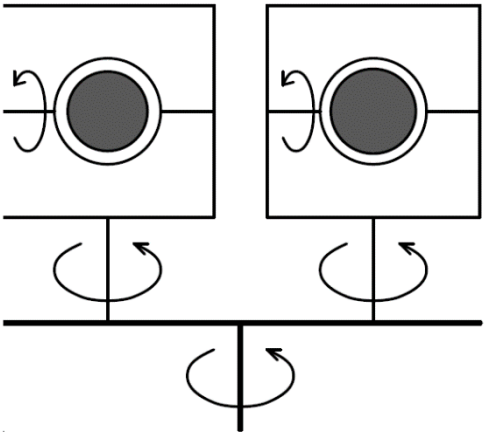


Figure 2. Independent Gun-Turret Model for a Sensor Platform (Murray, 1992).

As mentioned previously, the ability of each camera to tilt independently is probably a drawback of the independent gun-turret model, since such independent camera tilt is not necessary for three-dimensional stereoscopic vision. Further, if the two cameras tilt up and down at different angles then they will no longer necessarily converge on a single point. It may be difficult to have two separate motors simultaneously tilt each camera up and down the same amount, meaning this additional degree of freedom could do more harm than good.

As Murray et al. reported, "We have been unable to find overriding benefits for [the common elevation or independent gun-turret] configuration in terms of machine vision […], but the need to couple the two elevation axes by control in the gun-turret configuration, so that the optic axes remain coplanar, is a distinct disadvantage, and therefore in our work we have adopted the common-elevation platform" (Murray, 1992).

Actuators

There is a large range of actuator technology available for providing and controlling movement in robotic systems. In actuating robot "eyes" to mimic the movement of human eyes, the actuator should be small and compact, and capable of movement comparable to that of the human eye. Some actuators that have been used to successfully actuate human eyes-inspired robot eyes are piezoelectric, shape memory alloy, and pneumatic. DC and servo motors have also been used, but tend to be disregarded when precision, small size and speed are priority factors. The use of piezoelectric, shape memory alloy, and pneumatic actuators will be reviewed in this section.

Piezoelectric Actuators:

A piezoelectric actuator (also called piezo actuator) is a solid-state actuator which converts an electrical signal directly into precise, linear, physical displacement with virtually unlimited resolution (APC International, Ltd., 2002). Piezoelectric actuators are small-scale, stiff, load-bearing and stackable actuators. They are currently the most commercialized and understood technology in the smart actuator market (Trease, 2001).

Due to their precise movement capabilities and resistance to deterioration, piezo actuators are used in a variety of industrial, automotive, medical, aviation, aerospace and consumer electronics applications (APC International, Ltd., 2002). Piezo actuators are found in precision knitting machinery, braille machines, auto focusing mechanisms in microphone-equipped video cameras and mobile phones, and scanning probe microscope heads.

Using piezoelectric actuators, several researchers have been able to create robotic vision systems that replicate human eye muscle motion in order to actuate cameras. One such robotic system, shown in Figure 3, was created at the Georgia Institute of Technology. This lightweight, high speed robot actuates a camera with one degree of freedom, rotation about the horizontal axis (Goodwin, 2012).

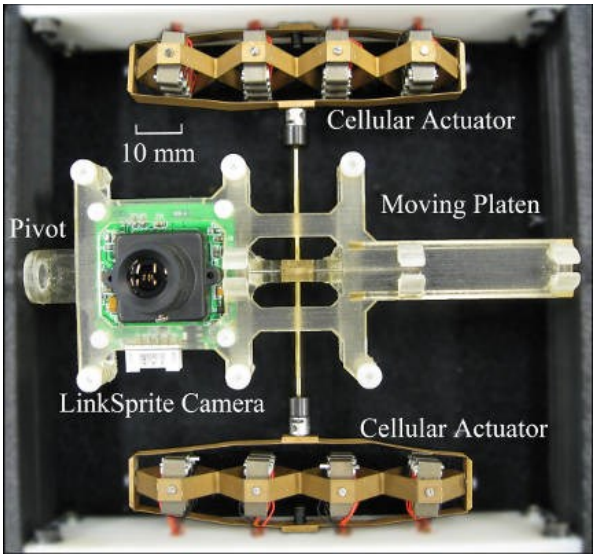


Figure 3: Piezoelectric Actuators used to actuate a Camera at Georgia Institute of Technology (Ueda, et al., 2010)

In order for the robot to be truly bio-inspired, it was determined that the actuators should possess properties similar to those of biological actuators. Actuators made from piezoelectric materials have an operational principle similar to that of human muscle. Human muscle expands and contracts due to neural impulse, while piezoelectric materials respond to electrical impulses (Goodwin, 2012).

To compensate for the small strain characteristic of the piezoelectric material, nested rhombus mechanisms were used to produce exponential strain amplification. As shown in Figure 4, the horizontal displacement of the piezoelectric material (grey) is magnified in the vertical displacement of the rhombus due to the geometry of the rhombus. If these rhombuses are nested appropriately, as depicted in Figure 5, the amplification effect can be significantly increased (Ueda, et al., 2010).

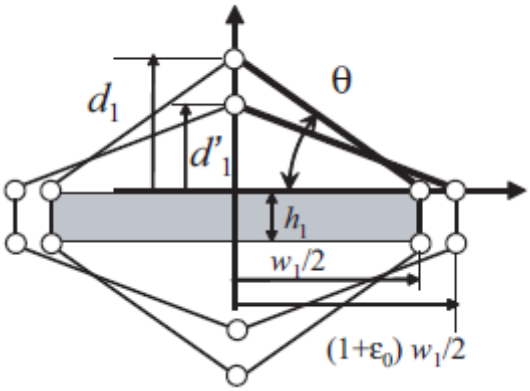


Figure 4: Strain Amplification Principle of Rhombus Mechanism (Ueda, et al., 2010)

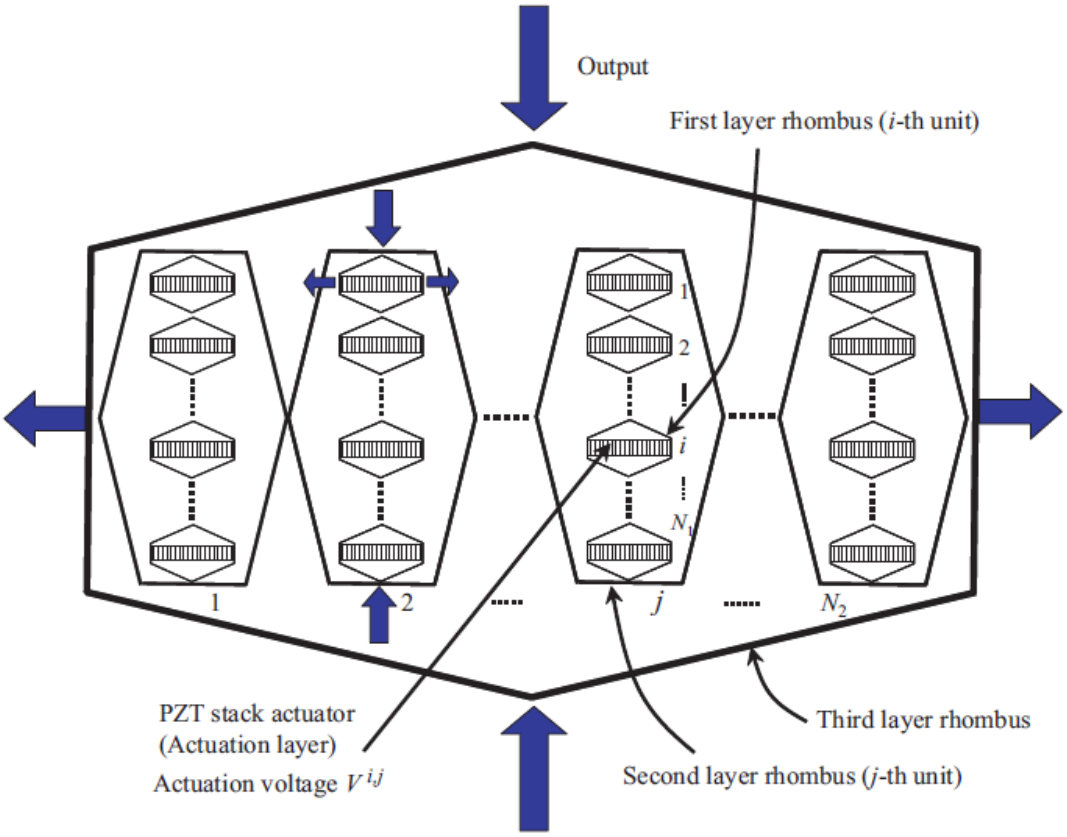


Figure 5: Significantly Increased Strain Amplification from Nested Rhombus Mechanisms (Ueda, et al., 2010)

Researchers at Institute of Applied Mechanics at the Technische Universität München also used piezoelectric actuators to create a robotic camera which mimics the motion of human eyes. This design focused on speed, and the piezoelectric actuators, visible in Figure 6, allowed the camera to reach speeds of 2500 degrees per second (human eyes can reach speeds of approximately 500 degrees per second). The piezoelectric actuators allowed the robot to be lightweight and fast. The actuators

provided high speed, acceleration and force with a small size, eliminating the need for a gear box (Guizzo, 2010).

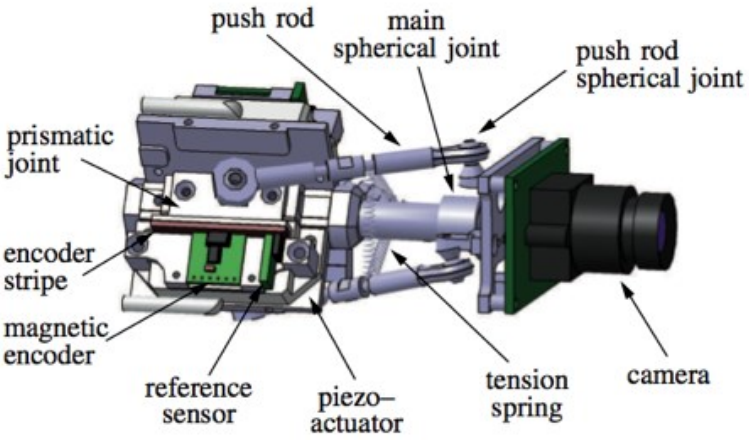


Figure 6: Superfast Robotic Camera Model (Guizzo, 2010)

The design consists of a parallel system of 3 piezos actuating a 3 degree of freedom system. The piezo actuators transmit their movement to a prismatic joint, which in turns drives small push rods attached to the camera frame. The rods have spherical joints on either end, and this kind of mechanism is known as a PSS, or prismatic, spherical, spherical, chain (Guizzo, 2010).

Shape Memory Alloy Actuators:

Shape-Memory Alloys (SMA) are low-stiffness, high-displacement, thermally-driven actuators. SMAs provide greater strain than other smart actuators and are simple to activate, requiring only the material and an electric current source. However, their thermal activation results in cooling-dependent response time, making SMA typically slow and unsuitable for high-frequency applications. In addition, they have very high hysteresis (Trease, 2001).

Researchers at the Department of Mechanical Engineering at the University of Alberta, Canada decided that SMA was the best actuator for their human prosthetic eye design. The actuator had to satisfy requirements for biomimicry, simplicity and optimization. SMA was chosen over traditional systems such as solenoids, electric motors, and hydraulic and pneumatic cylinders, as well as other smart actuators because of its expansion and contraction motion and simplicity, making it best for mimicking human eye muscles. It was also small enough for the design, and produced negligible vibration (Wolfe, et al., 2005).

Springs of SMA were used to significantly increase deflection as the effective length was increased. The significant increase in deflection came with a reduction in output force. However, this reduction of output force was negligible, due to the small forces required to rotate the eye. These SMA springs were attached to the ocular element of the prosthesis in the same manner that muscles are attached to the eye, with a system of agonist and antagonist pairs to pull the eye from opposite sides, as shown in Figure 7 (Wolfe, et al., 2005).

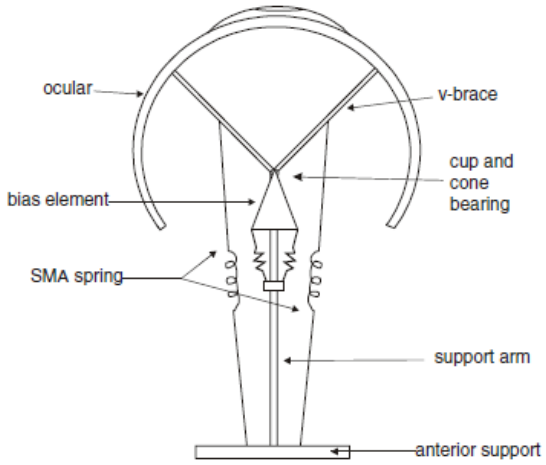


Figure 7: Schematic of the SMA Actuated Orbital Eye Prosthesis (Wolfe, et al., 2005)

Pneumatic Actuators:

Pneumatic actuators commonly involve a piston driven by pressurized gas (Hollerbach, 1992). Pneumatic actuators have been used to create Pneumatic Artificial Muscle (PAM), also known as air muscle. A PAM is a simple pneumatic device developed in the 1950s by J.L. McKibben. Like biological muscles, air muscles contract when activated. They provide a reasonable working copy of biological muscles that researchers can use to study the design of biological systems from an engineering point of view (Iovine, 2000).

PAMs were used in the State Key Laboratory of Fluid Power Transmission and Control at the Zheijiang University, China to actuate a 3 Degree of Freedom novel humanoid robot eye. After considering several types of actuators, PAM was found to have the most advantages for the robot because it is small, light, simple, user-friendly, soft, easily controllable and powerful. In addition, the contraction ratio and force-displacement function of PAM were found to be similar to those of human muscles (Wang, et al., 2008).

Six PAMs were used as Extra Ocular Muscles for the rotation of the eyeball, as seen in Figure 8. They were inserted in different positions on the surface of the spherical eyeball in such a way that cooperative stretching and shrinking of the PAMs resulted in 3 controllable degrees of freedom of the robotic eye muscles (Wang, et al., 2008).

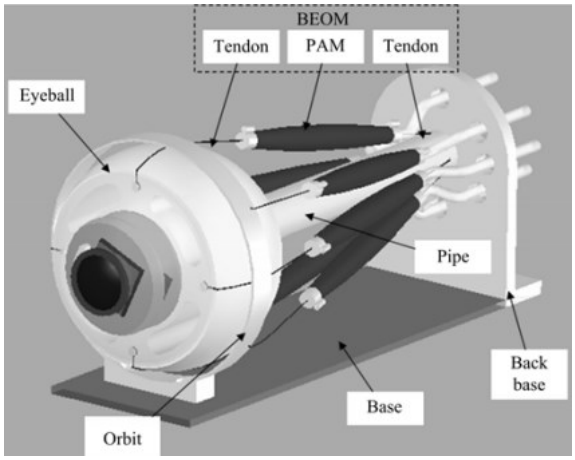


Figure 8: Prototype of Humanoid Robot Eyes using Pneumatic Artificial Muscles (Wang, et al., 2008).)

2.2.3 Image Processing

Robot vision systems must also handle the non-physical aspect of the vision system, specifically the sensory data obtained by the system. These data form an image from which information can be extracted.

Remote Sensing

Remote sensing is defined as the acquisition and measurement of information about certain properties of phenomena, objects, or materials by a recording device not in physical contact with the features under surveillance. In terms of image processing, remote sensing typically refers to technologies for recording electromagnetic energy that emanates from various objects. The different levels of electromagnetic energy associated with different objects provide a way for objects to be distinguished and identified (Khorram, et al., 2012).

The two main types of remote sensing are passive and active. Passive remote sensors detect natural radiation emitted or reflected by the object or surrounding area being observed. The most common source of radiation measured by passive sensors is reflected sunlight. Examples of passive remote sensors include film photography, infrared, and radiometers (Abdulrahman, 2010).

Active remote sensors first emit energy in order to scan objects and areas then sense and measure the radiation reflected from the target. Radiation Detection and Ranging (RADAR) and Light Detection and Ranging (LIDAR) are examples of active remote sensors (Abdulrahman, 2010).

Remote sensing is used in robotics to gather information about the environment for transmission or image and data processing. Robots can use either passive or active remote sensors to achieve this purpose. Robots can be equipped with passive cameras to take pictures, or be equipped with an active sensor like a LIDAR to create a digital map of an area.

Three-Dimensional Reconstruction

Binocular vision uses images from two eyes to construct a mental image of a 3D scene. Stereoscopic cameras use two lenses with separate image sensors to mimic binocular vision. This allows accurate depth information to be gathered and an object or scene of interest to be visualized in three dimensions (Kyto, 2011).

While the human brain and the brains of other animals with binocular vision have evolved to combine and make sense of the images captured from each eye, stereoscopic cameras and other

binocular sensor systems face a problem of three-dimensional image reconstruction. Two-dimensional sensor data is captured from multiple points of view and this information is used to construct a three-dimensional image of an object or scene.

Triangulation:

If two angles of a triangle and the length of the side between those angles are known, then the lengths of each of the remaining sides and the location of the third point of the triangle can be determined. Triangulation is the process of determining the location and depth of this third point using this information.

Stereoscopic cameras use triangulation to determine the location and depths of points on objects or parts of a scene. Specifically, the location and depth of a point of focus is calculated using the distance between the two cameras, called the baseline, and the direction of the point from each camera (Kyto, 2011). Since the baseline is usually fixed it can be easily known, but when it is not fixed it must be actively measured. When a stereoscopic camera has two fixed cameras that always have the same orientation, the task of determining the direction to a point of interest from each camera becomes the fairly simple task of analyzing the location of the point in each image.

However, when the cameras are not fixed, the problem of determining the direction of a point of interest from each camera becomes slightly more complicated, since the orientation of each camera changes. Consequently the direction to a point of interest from each camera depends on two changing variables: the orientation of each camera and the location of the point of interest in each image.

When designing a stereoscopic sensor platform that has cameras that change orientation and that uses triangulation to determine the location of points in a scene, it may be desirable to simplify the task of determining the location of a point. This can be accomplished by pointing each camera directly at the point for which one wants to calculate the depth. In this way the point of interest will always be in the center of each image. Consequently, the direction of each point of interest from each camera will only be dependent on one changing variable, the orientation of each camera, rather than both the orientation of each camera and the location of each point in the scene.

Correspondence Problem:

The correspondence problem is the problem of determining which parts of one image correspond to which parts of another image (Murray, 1992). While humans have specialized brain regions to do this well without any conscious thought, a stereoscopic camera or other binocular three-dimensional visualization device must be designed with a specific methodology in mind to solve this problem.

There are two basic methods that can be used to find the correspondence between two images. The first method, called the correlation method, involves checking to see if one part of one image looks like another part of the other image.

The second method of finding correspondence between two images is called the feature-based method. This method involves looking for certain features in one image and then attempting to identify whether the same features are present in another image.

Evidently, both methods are very software-intensive. They both involve analyzing the images captured from each camera in a specific way. From the perspective of the design of the camera platform, the main way to make three-dimensional visualization easier is to simultaneously point each camera directly at the object of interest. Finding the correspondence between each image is then straightforward, since the center of one image corresponds with the center of the other image.

3 Design Goals and Specifications

In the following sections, a goal statement for the project is provided as well as the design specifications that the team developed to aid in design evaluation.

3.1 Goal Statement

The goal of this project was to develop, analyze, construct, and test a mechanical platform for a binocular robotic vision system that is capable of pointing two eye components in a coordinated manner at an individual point of interest in a scene through mechanically coupled pan, coupled tilt and coupled convergence movements.

3.2 Design Specifications

The design specifications were divided into three overall categories:

(1) Humanoid / Social Function

(2) General Function (Kinematics)

(3) Manufacturability / Durability

Each design specification and its corresponding rationale are listed below, in the following format:

- Design specification
    - Rationale

3.2.1 Humanoid / Social Function

In order to design the mechanical platform for use in social robot applications, many of the design specifications were selected with the capabilities of the human vision system in mind.

- Must be capable of panning, tilting and converging each “eye” component about its respective center.
    - It is crucial that the robotic vision system can reproduce this fundamental ability of human eyes for socially acceptable functionality in a social-robotics application.
- Must be able to make two “eyes” simultaneously aim at single points within a certain range of vision (defined in later specifications).
    - This is a fundamental function of the design. Since human eyes always converge on the same point when in use, it is important that the “eyes” in this platform also simultaneously converge on points in their field of view.
- Should be able to rotate each “eye” with a speed of at least 500 deg/sec, and angular acceleration of at least 25,000 deg/sec^2.
    - These quantities represent the pan velocity and acceleration capabilities of the human eye (Villgrattner, 2009). In addition to mimicking the abilities of the human vision system, this specification is important because the angular velocity and acceleration capabilities of the “eyes” limit how well each eye can carry out smooth pursuit while tracking an object.
- Should have less than 5° of angular twist when returning to its original state (looking forward).
    - Due to the non-commutativity of rotations, after each rotation of the eye some angular twist may occur. In order to prevent mechanical failure due to this angular twist and

keep the eyes looking as human as possible, there should not be a lot of angular twist accumulated when the eyes' gaze returns to its original state.

3.2.2 General Function / Kinematics

The General Function / Kinematics category of design specifications is further divided into the following three groups:

(1) Basics of Camera/Sensor "Eyes"

(2) Data on "Eye" Directions, Positions, Velocities and Accelerations

(3) Field of Vision

Basics of Camera/Sensor "Eyes"

- Must have a means to attach two cameras/sensors/lasers ("eyes").
    - While a platform could be constructed with one specific type of "eye" permanently attached, the goal of this project was to design a mechanical platform capable of controlling the motion of a variety of "eye" components. Consequently, the platform must have a means to attach these various "eye" components.
- Must be compatible with "eyes" that are larger than 1 cm by 1 cm by 1 cm and smaller than 5 cm by 5 cm by 5 cm.
    - If the "eyes" serve a social robot application, the "eyes"—which communicate with the human user through gaze and eye contact—should be a realistic size. The lower limit is in place for manufacturing reasons (if cameras are implemented in the eyes, very small cameras will be difficult to work with); the upper limit is in place to make the eyes realistic.
- Must be capable of moving "eyes" that weigh 100 g or less.
    - Due to inertial constraints, the platform may not be able to handle "eyes" with very high masses. However, it is important that the platform can be used to control the motion of "eyes" that weigh up to 100 g.
- Must have "eyes" mounted such that the centers of rotation of the eyes are between 5 cm and 13 cm apart horizontally.
    - The values for this number are also based on the average interocular distance for humans (approximately 2.5 cm) (Gordon, 1988). From a social function standpoint, a human interacting with the robot must be able to look at both eyes at once, i.e., make eye contact as humans do with one another. The platform may be scaled up from the size of actual human dimensions for ease of manufacturing.
- Must be capable of converging "eyes" on points with a pan resolution no greater than 2 cm, a convergence (depth) resolution no greater than 5 cm, and a tilt resolution no greater than 1°.
    - To clarify, the above means that the platform must be able to change the focus point pan position by increments of 2 cm or smaller, the depth by increments of 5 cm or smaller, and the tilt by increments of 1° or smaller. These values were chosen because they are small enough to allow the "eyes" to easily focus on objects the size of hands or faces during social robotic applications, but not so small as to be infeasible to achieve given available resources.

Data on "Eye" Directions, Positions, Velocities and Accelerations

- Must determine the directions the "eyes" are pointing relative to a predetermined initial position within 1° of accuracy.
    - Processing the camera/sensor data from the multiple "eyes" to achieve 3D visualization is not possible without knowing the directions the "eyes" are pointing. The greater the accuracy here, the better, but with the given budget greater accuracy than this cannot reasonably be demanded.
- Ideally should determine the velocity of the "eyes" and the acceleration of the "eyes."
    - These two values could technically be calculated as time derivatives from the position information required by the previous design specification. However, if the position information was not gathered frequently enough then the calculation of the velocity and acceleration could have an unreasonably large error. Further, the time delay of these calculations and/or the processor power needed to perform the calculations may be reason enough to include an accelerometer. Data on the actual velocities and accelerations of the "eyes" would be used to test our velocity and acceleration design specifications for the "eyes."

Field of Vision

- Should be able to aim (tilt) the "eyes" 45° above and 45° below the horizontal plane.
    - For a social robot application, the "eyes" themselves should have a vertical tilting range of motion that is similar to that of humans.
- Should be able to converge the "eyes" at depths greater than or equal to 0.5 m and less than or equal to 2.5 m.
    - For a social robotic application, this is a reasonable operating range.
- Should be able to pan the eyes at least 45° left and right of straight ahead.
    - Again, this is a reasonable operating range for a social robotic application.

3.2.3 Manufacturing / Durability

- Must have a material cost of less than $800 in total.
    - If the entire platform and hardware costs too much, it becomes impractical for most widespread use or purchase. Also, the project budget is limited.
- Should be able to fit within a space that is 45 cm x 45 cm x 45 cm. While a compact design is preferred, some specifications have been relaxed to reduce costs.
    - The allowable interocular distance stated previously is about double that of a human. The 45 cm limit selected here is approximately the size of a human head scaled up by the same factor.
- Should weigh no more than 8 kg.
    - The mass of the entire platform should be low so that it is portable. Note that while it will probably be necessary to make the masses of moving parts in the platform low in order to meet all of the above design specifications, there is no need to set a limit on any individual part of the platform.

4 Design Concepts

During the design process, numerous designs were considered to achieve mechanically coupled pan and coupled convergence eye movements. The tilt mechanisms were designed separately in such a way that they could be used in combination with each pan/convergence mechanism. The two main pan/convergence concept designs are described below.

4.1 Prismatic-Revolute (PR) Design

The initial concept of the prismatic-revolute design was based on a pair of eyes controlled by two servo motors, as seen in Figure 9. This design works through mechanically coupling the tilt in each eye and the pan in each eye with respective servos. This mechanism works similarly to the initial PR design concept shown in Figure 10, where motor 1 actuates links L1, L2, and L3 to drive pan in each eye, and motor 2 actuates links L4, L5, and L6 to drive tilt. Platforms 1 and 2 are not separated in the inspiration model, but are instead just one solid platform. Since pan and tilt were not sufficient for the completion of our design, a way of adding a convergent element to the existing setup was sought.

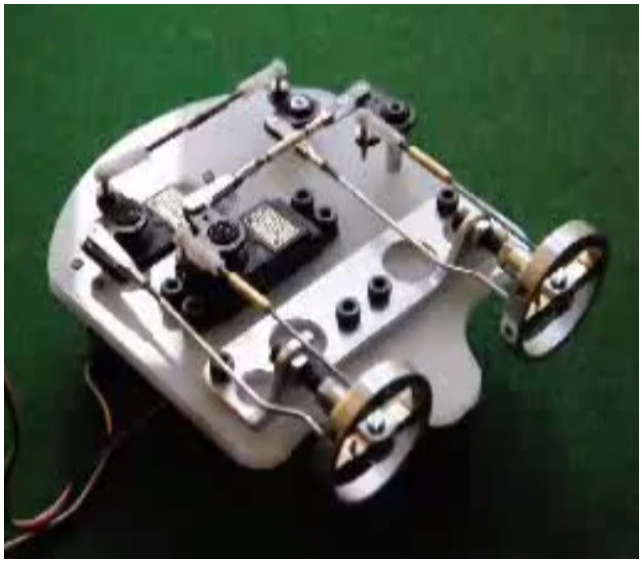


Figure 9: Prismatic-Revolute Design Inspiration (Thomas, 2009)

The first iteration of the PR design separated the two existing actuators onto separate platforms, and can be seen in Figure 10. Platform 2, which contains the mount for the pan actuator, would be movable, rotating around a shaft in the middle of the platform to induce convergence to the eyes. Platform 1, which supports the tilt mechanism, would remain unchanged, staying fixed to ground. Moving the pan actuation onto a separate movable base adds the third degree of freedom (convergence) necessary to our design.

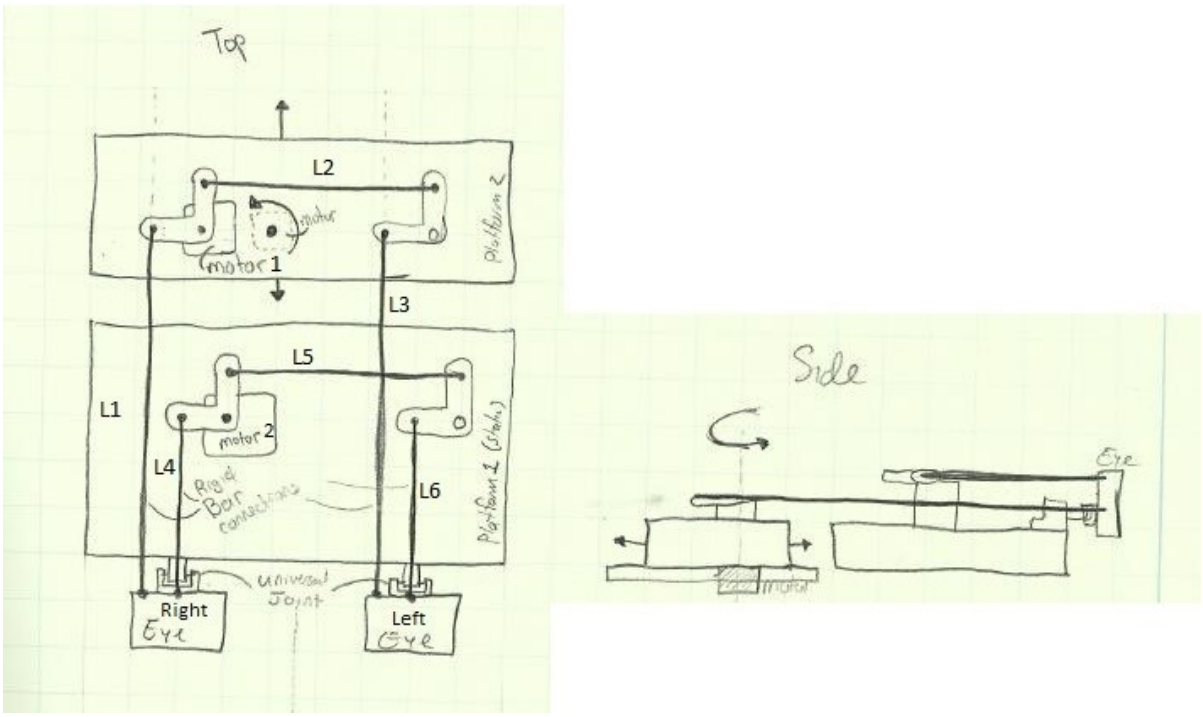


Figure 10: Initial PR Concept Design

Because of the high inertia that needs to be overcome to move the entire platform, the second iteration of the PR pan and convergence mechanism retains the single fixed platform and changed the actuation from the initial linkage to a single solid bar which can rotate about its midpoint, and can be seen in Figure 11. The joints connecting the actuator to the eyes were also moved from the right side of each eye to the insides of each eye. As the bar rotates about a vertical axis at its center, it turns the eyes an equal amount in the same direction, inducing pan. As the bar translates back and forth, it causes the eyes to also move the same angle, but in opposite directions, which induces the convergence. The empty standoffs in the rear end of the mechanism are for the tilt component, which is not included in the image. An advantage of this PR design is both pan and convergence mechanisms can be approximated by four-bar linkages, as either a fully pin-jointed four-bar pan or a slider-crank for convergence. Another advantage to this design is the high speeds at which the eyes may be rotated, which is desirable for this project. However, there is a very small range of actuation required to move the eyes, given the desired physical component size constraints, so very precise servos would be required to achieve an optimal resolution.

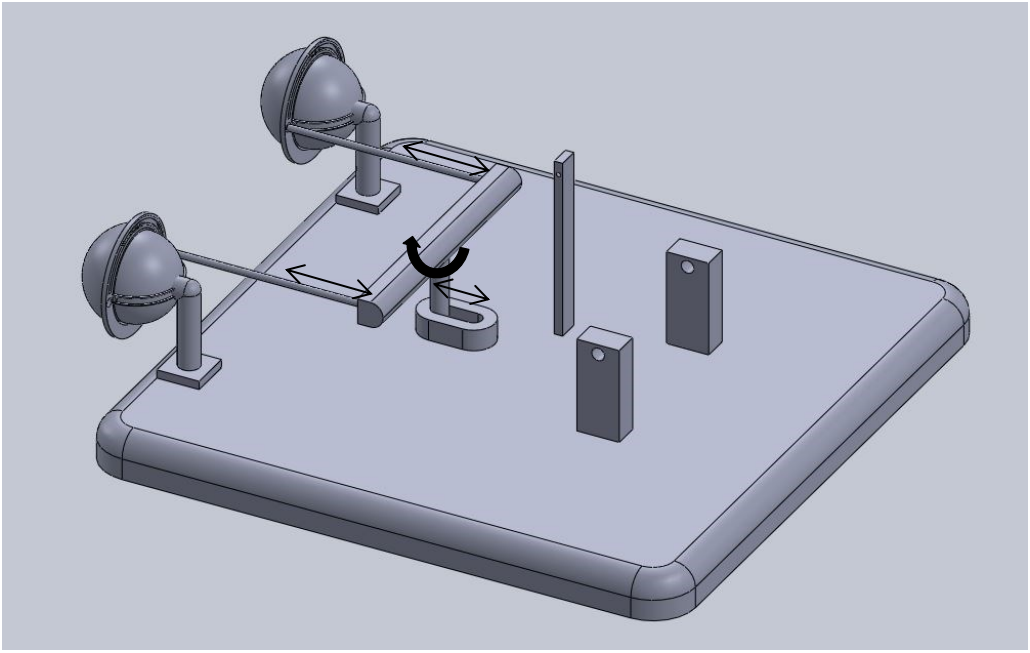


Figure 11: PR Second Iteration

4.2 Prismatic-Prismatic (PP) Design

The second main concept design, the Prismatic-Prismatic (PP) design, achieves mechanically coupled pan and coupled vergence movement of the two “eyes.” As shown in Figure 12 the design utilizes two motors to achieve linear actuation of a control bar link in the lateral (x) and proximal (z) directions. The control bar is pin-jointed to a slider block at each end, which slides in a slotted slider link. These two slider links are pin-jointed to the base platform (ground). Each “eye” component (with tilt mechanism, not shown) is mounted to one of the slider links such that “eye” is allowed to tilt. The rotation of the slider links about their pin joints to ground results in pan and convergence motion of the “eyes.” The centers of rotation of the “eyes” remain centered on the axis of the slider link pin joints to ground; therefore the distance between the pin joints along the base is equal to the baseline between the “eyes.”

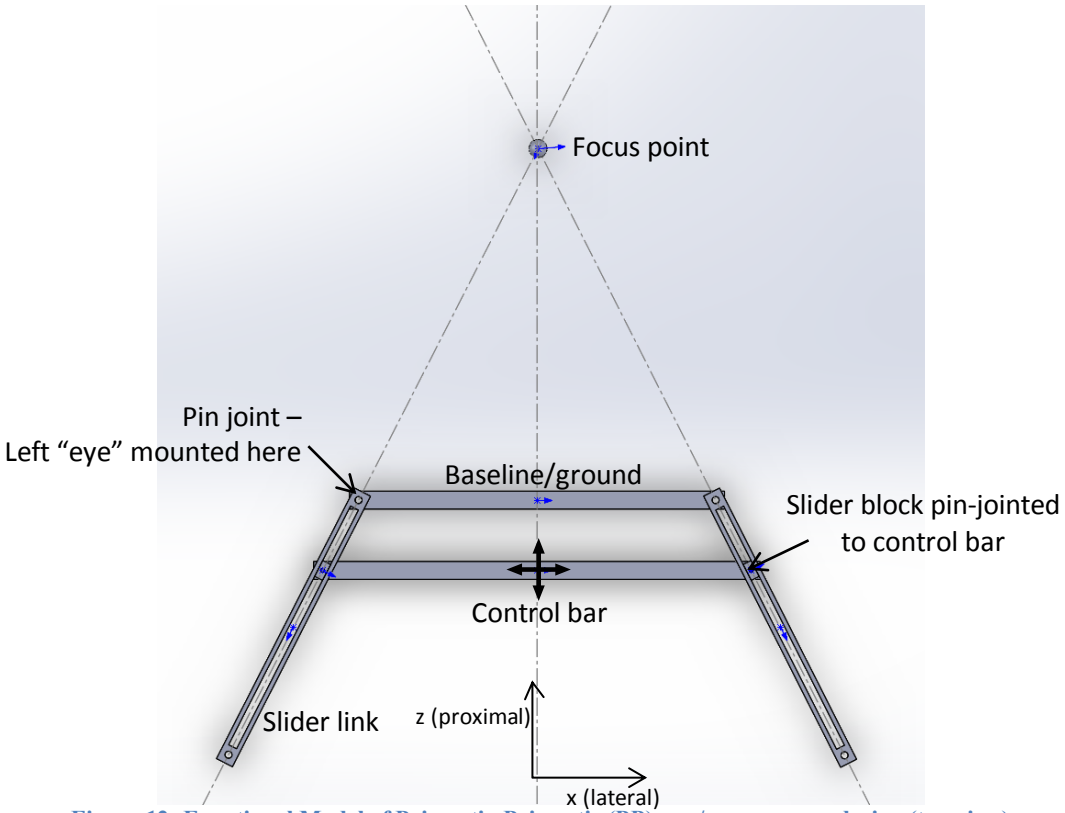


Figure 12: Functional Model of Prismatic-Prismatic (PP) pan/convergence design (top view)

In order for the line of sight of the “eyes” to intersect and converge on a single point in front of the mechanism, the control bar must be longer than the baseline. As the control bar translates proximally, the “eyes” carry out synchronized vergence movements: moving the control bar towards the baseline causes the eyes to focus on a point closer to the eyes (Figure 13a); moving the control bar away from the baseline causes the eyes to focus on a point further away (Figure 13b). As the control bar translates laterally, the “eyes” pan: moving the control bar to the left causes the focus point to move to the right (Figure 13c), and moving the control bar to the right causes the focus point to move to the left. The control bar is not allowed to rotate—it remains oriented parallel to the baseline at all times.

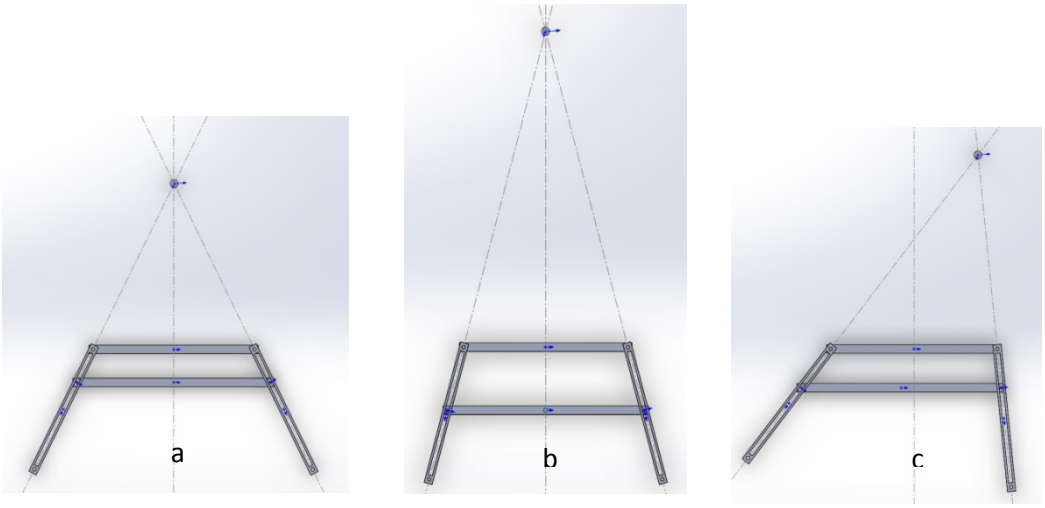


Figure 13: Top view of the PP Design in Different Positions, demonstrating Pan and Convergence
(a) Mechanism in an arbitrary initial position, with focus point centered between the eyes; (b) translating the control bar proximally causes the depth of the focus point to change; (c) laterally translating the control bar pans the focus point.

If the lines of sight of the "eyes" are parallel to their respective slider links, pure lateral translation of the control bar will pan the focus point and maintain a constant proximal distance from the focus point to the baseline. This was confirmed in a SolidWorks motion analysis (see Figure 14). The fact that the convergence depth is decoupled from pan movements is a significant feature for control purposes in that it simplifies calculation for actuation given a focus point location. It has the limitation that the "eyes" cannot simultaneously be aimed straight ahead at a focus point of infinite depth. Note, however, that the design specifications listed previously do not require infinite range.

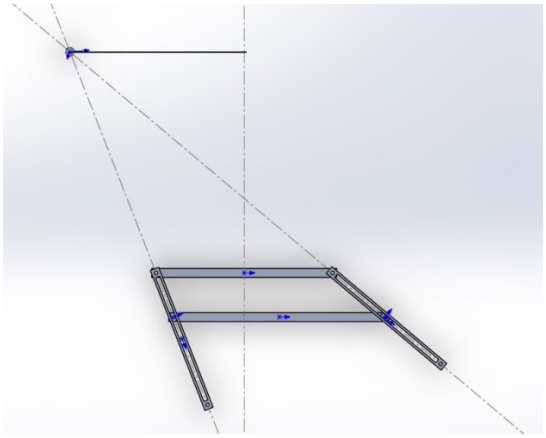


Figure 14: PP Pure Lateral Translation

5 Design Selection

To effectively compare the PR and PP designs, numerical analyses were conducted on the designs. Various equations were formulated to relate design parameters and dimensions, and the values of certain parameters were constrained to specific input values by our design specifications. The initial desired design specifications included range of vision, resolution, and eye speed. Output parameters included the dimensions and overall workspace of the mechanism and the required motor angular velocities. The values obtained for the output parameters can be compared across the two designs.

5.1 PR Design

Previous examination of the PR design revealed the design's greatest flaw to be its proximal depth resolution, so this was evaluated first. Given the proximal focal range z_{min} = 0.5 m and z_{max} = 2.5 m, the link lengths and orientations must be optimized to maximize the number of steps within that range. An initial iteration had the coupler link b = 12 cm, and eye radius c = 3 cm, see Figure 15. The length "a" does not affect the convergence calculation.

Using typical stepper motor specifications (200 steps/rev), and assuming the pinion radius to be 3 cm, point "2" moves 0.942 mm for each step of the motor. This distance per step, in addition to the initial link lengths, results in an average angular step dθ of 1.8°, or a proximal focus point change of 65cm/step.

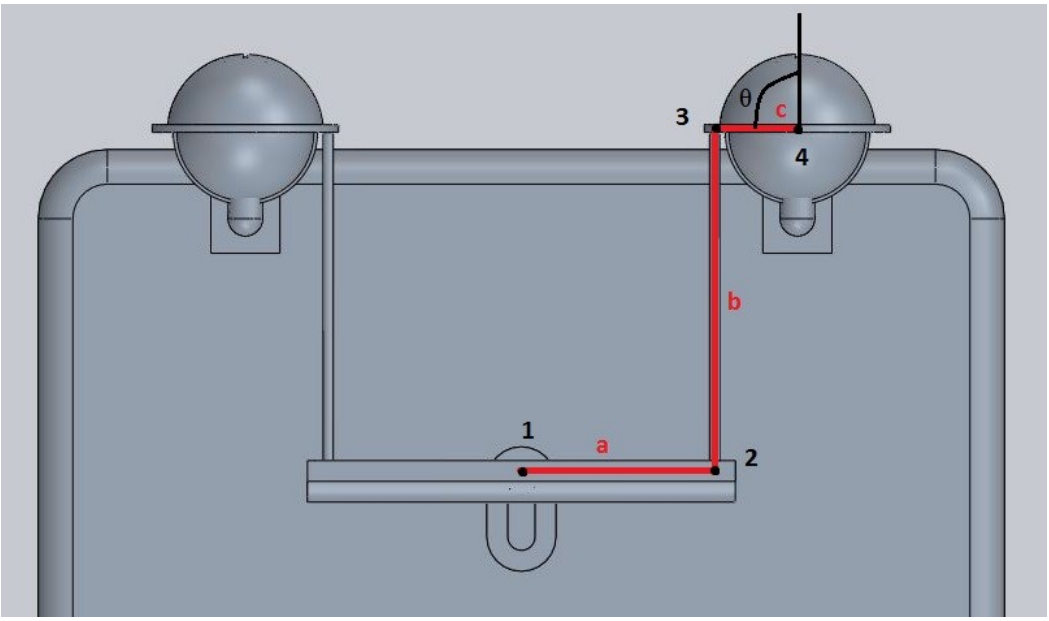


Figure 15: PR Pan/Convergence Linkage

With a 12.5 cm fixed baseline dimension, and the desired proximal focal range, the angles of θ varies from 82.9° to 88.6°, a range of only 5.7°. Because the range of angles is so small, and the angular step is so large, there are only 3 steps over the entire proximal range. This is an unsatisfactory result.

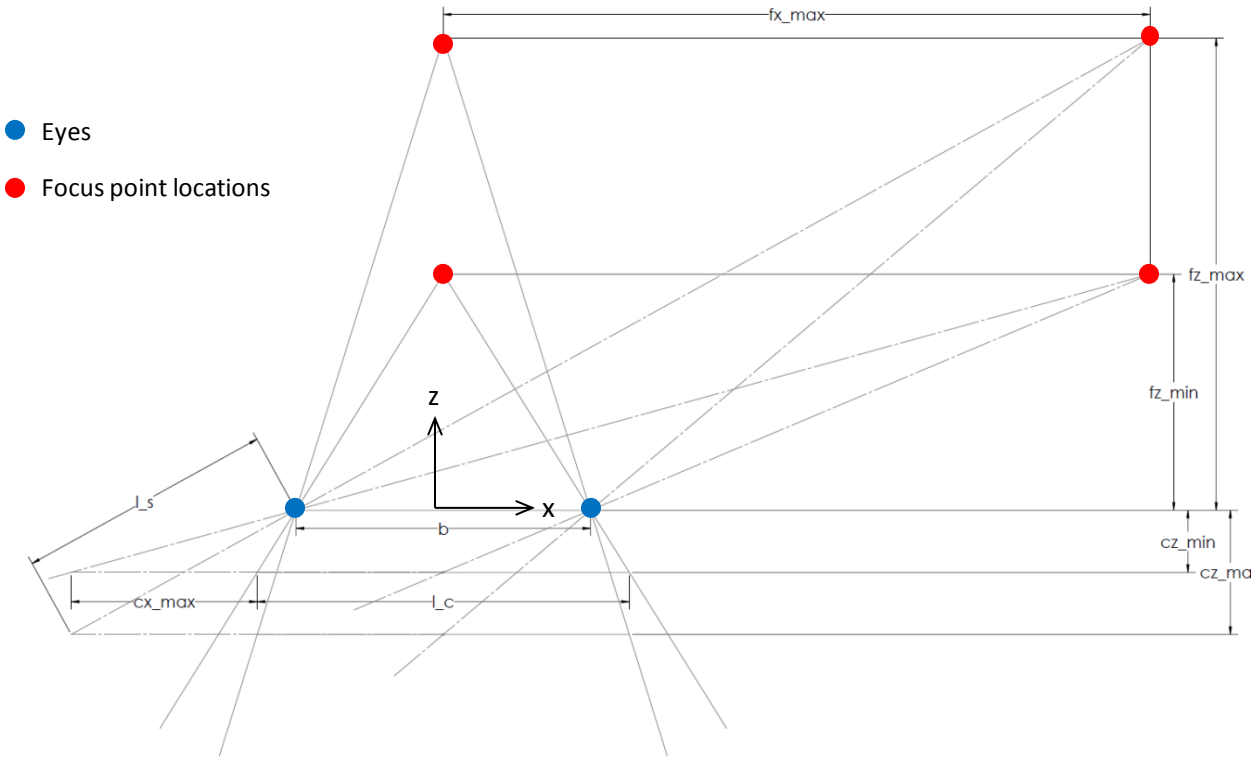
In an attempt to improve the resolution, the link lengths and positions were varied. The lengths of links "b" and "a" were increased and decreased, resulting in different positions of joint "2". None of the individual alterations nor combined changes resulted in an increase in the number of steps over the

range. Since the design cannot be improved by dimensional changes, the only other solution to increase the resolution would be to utilize a system of gears.

A gear ratio of 1:5 would result in a proximal step size of 12.5 cm/step. It is a great improvement from the initial step size of 65 cm/step, but it is still unsatisfactory. Gearing down to 1:10 results in 6cm/step, which still does not meet the design specifications. A compound gear train would be required to further increase this ratio, but would result in the need for additional space for the gear train, as well as increased backlash between each pair of gears. Due to these considerations, the PR design was deemed impractical.

5.2 PP Design

Figure 16 shows the variable names that were used for several design parameters in the PP Design. The base coordinate system is at the midpoint of the eye baseline, and it is based on unconventional left-handed coordinate system found in image processing. Some of the values for these design parameters, such as the maximum focus point depth (fz_{max}), were chosen based on the design specifications. Such design parameters were labeled "input" parameters and are listed in Table 1. The rest of the design parameters, such as the maximum control bar depth (cz_{max}), were calculated based on the "input" parameters. These calculated "output" parameters are listed in Table 2.



Y axis is defined such that the positive direction is pointing out of the paper.

Figure 16: Diagram of PP Design Parameters

Table 1: PP Design Input Parameters for Design Selection Numerical Analysis

PP Design Input Parameters for Design Selection Numerical Analysis			
Parameter	**Description**	**Value**	**Justification**
b	Baseline length	12.5 cm	Distance between two eyes-chosen design specification
fz_{max}	Maximum convergence depth of focus point in z-direction	250 cm	Chosen based on social robotic application – max distance at which robot would communicate with a person
fz_{min}	Minimum convergence depth of focus point in z-direction	50 cm	Chosen based on social robotic application – min distance at which robot may interact with a person
Δfx	Pan resolution (smallest increment that focus point translates in x-direction)	2 cm	Picked based on social robotic application – this resolution would allow the robot to focus on facial features at the desired ranges of vision (fx and fz min/max)
Δfz	Convergence resolution (smallest increment that focus point translates in z-direction)	5 cm	Picked based on social robotic application – convergence resolution does not need to be as fine as pan resolution for the purposes of the robot looking realistic, but this resolution will allow robot to detect if a person is moving or communicating with head movements
ω_{pan}	Max angular velocity of focus point WRT center of baseline for minimum focal depth configuration	500 deg/sec	Saccadic human eye speed
$motor_{res}$	Number of steps in one revolution of the motor	200 step/ rev	Typical stepper motor resolution
fx_{max}	Maximum distance in x-direction focus point can be panned from centerline (straight ahead).	250 cm	Chosen based on social robotic application – approximate room size
cz_{min}	Minimum proximal distance of control bar from the baseline	2 cm	Arbitrary value - depends on the final design (tilt mechanism and control bar)

Table 2: PP Design Output Parameters Calculated from Input Parameters in Table 1

PP Design Output Parameters Calculated from Input Parameters in Table 1		
Parameter	**Description**	**Value**
W	Maximum dimension of the workspace in the z-direction	33.6 cm
l_c	Control bar length	13 cm
l_s	Slider length	11.29 cm
cz_{max}	Maximum proximal distance of control bar from eye baseline	10 cm
cx_{max}	Maximum lateral distance of control bar (left or right of center position)	5 cm
Δcx	Minimum lateral increment of control bar	0.08 cm
Δcz	Minimum proximal increment of control bar	0.2 cm
ω_{motor}	Maximum angular velocity of the pan stepper motor which actuates lateral control bar movement in x-direction	5.89 rev/s
r_x	Radius of pinion for rack x movement	2.5 cm
r_z	Radius of pinion for rack z movement	6.4 cm

From the numerical analysis, it was found that the PP Design is capable of satisfying the design specifications for range of vision, resolution, eye speed and workspace. The values obtained for parameters such as pinion radius, control bar increments, and stepper motor angular speeds were reasonable and achievable.

Appendix A: Generating System of Equations to Determine Input and Output Design Parameters gives detailed explanations of how the equations relating the input parameters to the output parameters were generated. Appendix B: Mathcad Initial Design Output Parameters Calculation shows the calculations of the output design parameters in Table 2 from the input design parameters in Table 1.

The motor speed, ω_{motor}, in Table 2 refers to that of the stepper motor which drives the control bar pan movement in the x-direction. While it was important that the eye pan speed meet the saccadic eye speed specification represented as ω_{pan} in Table 1, the convergence speed was left as a flexible parameter that could vary based on torque and velocity capacity of the motor.

The equations were based on the worst case scenario to meet the input design parameters. For instance, when calculating the pan motor speed necessary to achieve a saccadic eye speed of 500 deg/s, the eyes were assumed to be positioned at their maximum depth, fz_{max}, because the eye pan speed is slowest at this depth. In addition, the saccadic eye speed was considered as the average pan speed of the two eyes in the analysis for easier calculation. The average pan speed of the two eyes is defined as the pan velocity at the base coordinate system defined in Figure 16.

The Mathcad program created to do the numerical analysis on the PP Design can be used to iterate the outputs—for example, the program outputs a motor speed and pinion radii for specified desired inputs/dimensions (See Appendix B: Mathcad Initial Design Output Parameters Calculation). These generated values were used to determine an appropriate standard motor speed and gear radii;

these standard values were then used as the inputs of the program to obtain parameters such as pan speed and actual design dimensions that could be used with standard motors and pinions. Consequently, some of the initial design specifications will be met, exceeded or missed due to engineering trade-offs. Section 6.3.2 explains the final design specifications and parameters in detail.

5.3 Design Conclusion

In summary, the PP Design was selected over the PR Design because the PR Design's very limited convergence (depth) resolution. If both the PR and PP were designed with the same workspace size and similar eye movement speeds, it was found that the convergence resolution would be at least an order of magnitude better on the PP design than the PR design (Δfz = 5 cm for PP and is over 50 cm for PR).

6 Final Design

The CAD models for the platform were created almost exclusively in Creo Parametric. A few parts were conceptualized in SolidWorks, but were ultimately incorporated into the Creo model.

The platform design went through several iterations. Each iteration generated changes to the dimensions of the design due to realizations of constraints imposed by physical limitations and the availability of specific part sizes as well as refinement of the design specifications.

6.1 Iteration 1

The first CAD model of the PP design was created using primarily VEX robotics models acquired from the VEX robotics product website (VEX Robotics). VEX parts were selected because:

- Parts and CAD models were readily available: physical parts could be acquired from the WPI robotics lab, and complete product models and details from the VEX product website.
- VEX already had linear slides and hardware designed to mate with each other.
- Dimensions of the available parts could be used as preliminary dimensions for the model.

The pan and slider mechanisms each consisted of a slider, a slider track, a rack and pinion, and a stepper motor. The pan mechanism was mounted on the convergence mechanism by the mounting plate. The motions from the mechanisms were translated to the eyes by two sliders called the control arms. The initial pan/convergence mechanism CAD model, shown in Figure 17, was made entirely of VEX parts, with the exception of the motors and the controls pins, which were the pins that connected the control bar to control arms. This design generated 8 cm of proximal movement and 10 cm of lateral movement of the control bar. Everything mounted on the mounting plate moved with the convergence slider in the 12 in. slider track.

The issues that arose from this initial design were as follows:

- The pan mechanism was only supported at its center by the convergence mechanism slider, which could result in instability.
- The bending moments on the mounting plate could result in the misalignment of the bevel gears.
- The control pins were rather long, which could lead to more pronounced bending and tilting.
- In this design, the length of the control bar was dependent on the spacing of holes on the VEX parts and could not be fine-tuned to our design specification dimensions.

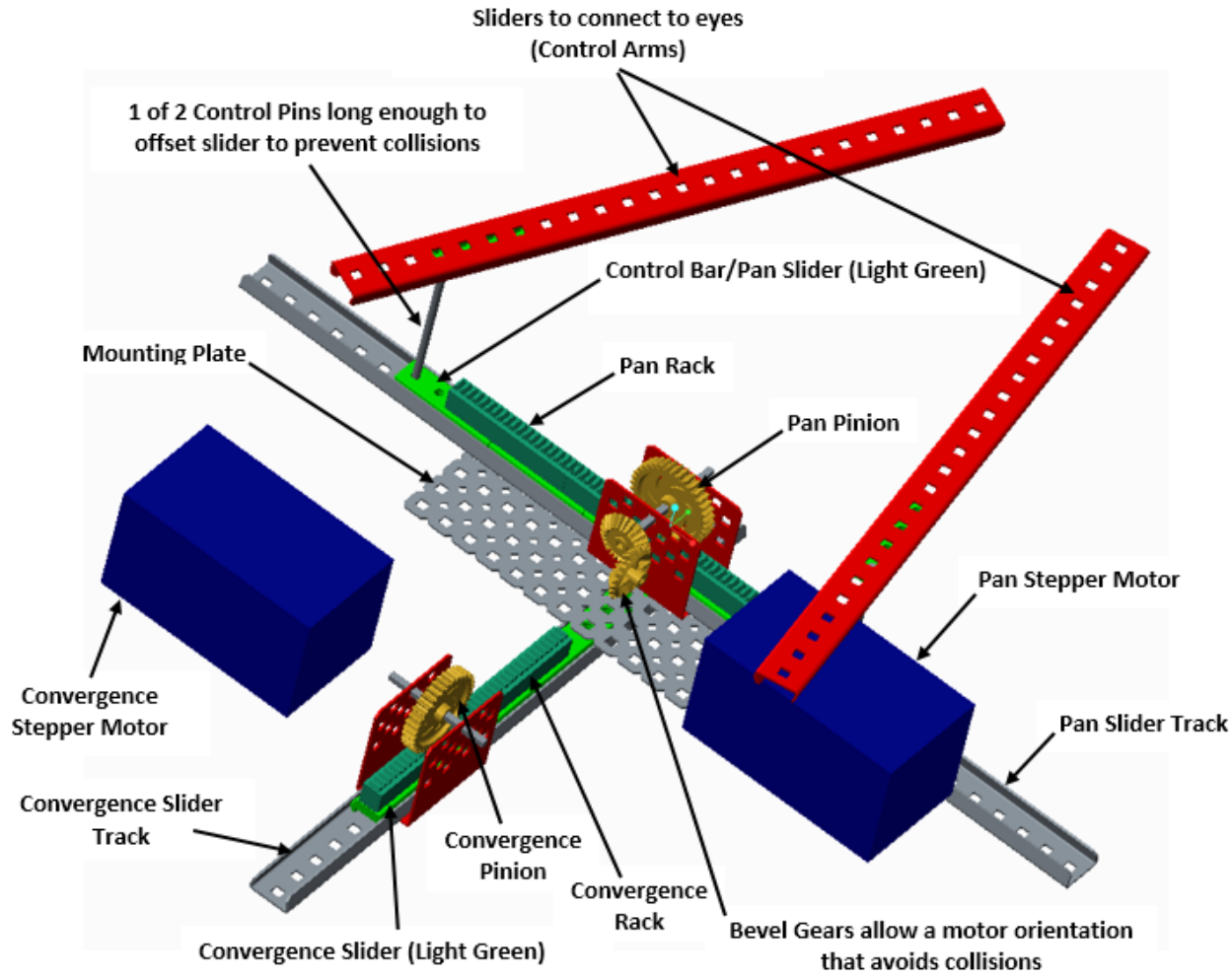


Figure 17: Initial Design of Platform Pan Convergence Mechanism

6.2 Iteration 2

The second pan/convergence design, shown in Figure 18, sought to address the issues of the first design. An additional slider was added to the convergence mechanism to increase the support for the pan mechanism. A control arch, detailed in Figure 19, was added to reduce the necessary length of the control pins (the pins that connect the control bar to the control arms). The control arch was to be a machined part, which would allow selection of the effective length of the control bar.

Additionally, the stepper motors were placed under the platform's base to avoid motor interference with moving parts above the platform base. Power was to be transmitted to the shafts with the use of the bevel gears, as shown in Figure 18.

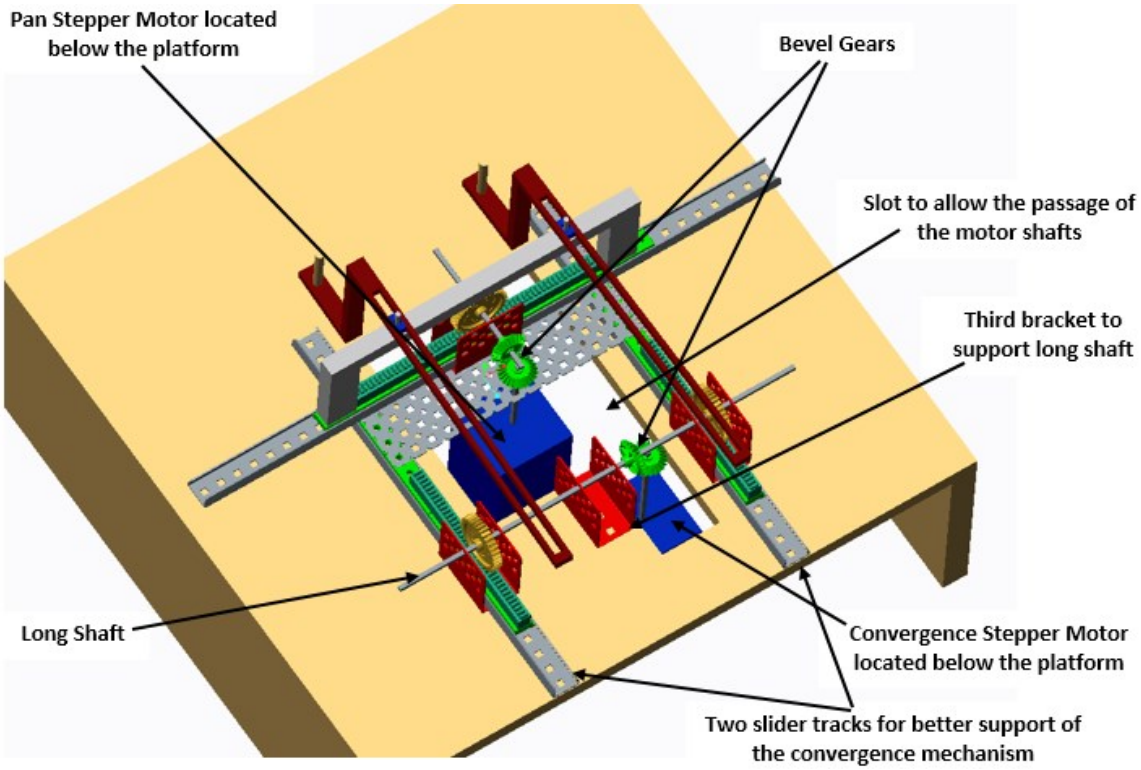


Figure 18: Second Design of Platform Pan Convergence Mechanism

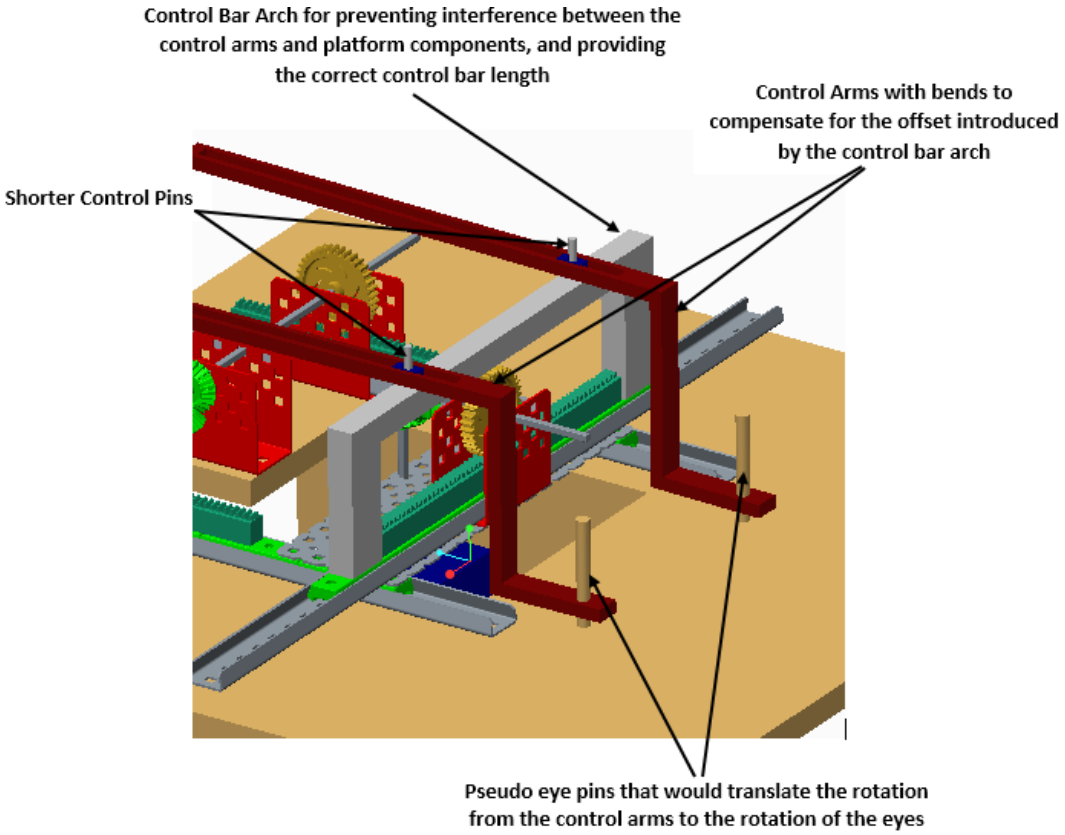


Figure 19: Adjustments to the Control Bar and Arm

6.3 Iteration 3: Final

The third iteration of the design was developed to eliminate the design problems encountered in previous iterations as listed below.

1. To eliminate the use of long shaft driven by convergence motor.
2. To reduce the length of offsets used to avoid interferences of mechanical parts and links.
3. To eliminate the use of bevel gears, reducing backlash and providing secure gear mating.
4. To simplify and reduce the number of mechanical parts.

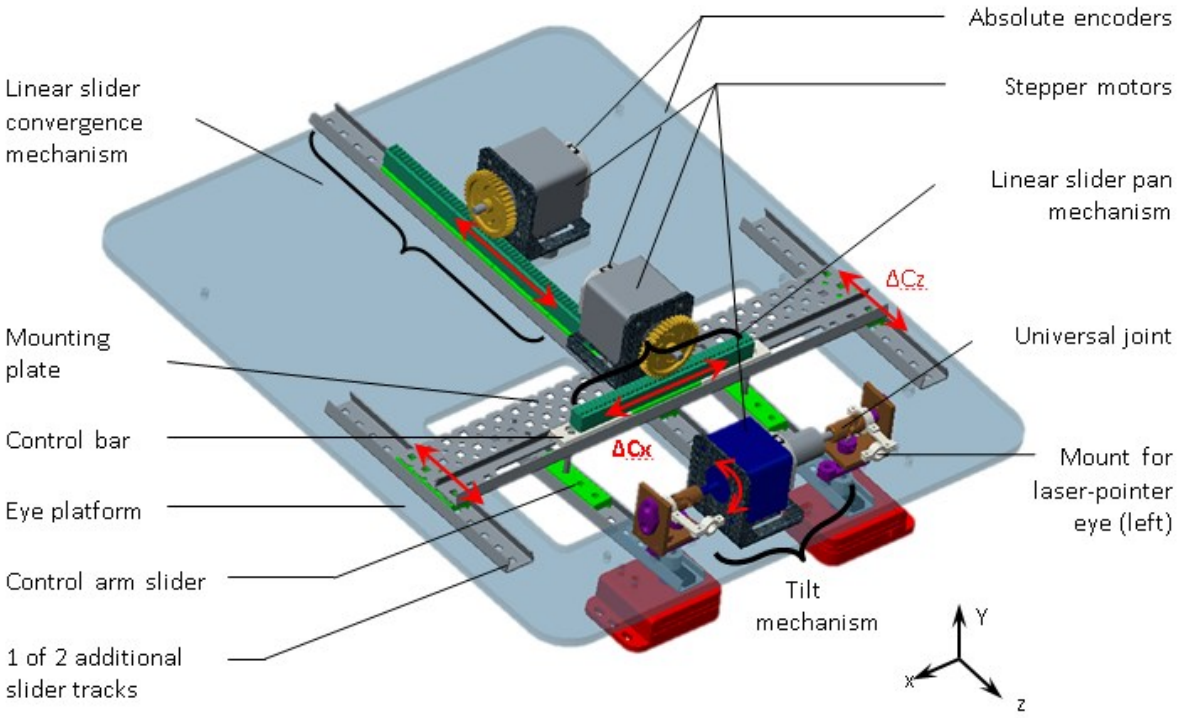


Figure 20: Completed Model of Platform

6.3.1 Design Description

The final design is comprised of three primary mechanisms: one for convergence, one for pan, and one for tilt. Each is described below.

Convergence Mechanism

The linear slider convergence mechanism consists of three tracks with inner sliders, a rack and pinion and stepper motor. The center convergence rack carries the plate to which pan motor and track are mounted. The plate is also supported by two additional slider tracks on each end to minimize bending and instability as the pan rack translates in the z-direction. The convergence track accommodates not only the length and translation of the convergence rack to locate the control bar at a maximum distance from the baseline but also the length of the pan motor so that there is no collision

between the pan and convergence motors when the control bar is moved further away from the eye baseline.

The convergence stepper motor is mounted on the platform via stepper motor mounting bracket. An absolute encoder is also mounted to the shaft of the stepper motor at the other side of the motor which does not drive the pinion.

Pan Mechanism

The pan mechanism consists of the pan mounting plate, linear slider pan mechanism (track, inner sliders, rack and pinion), stepper motor and control bar. The pan track accommodates the translation of the control bar to pan the focus point from extreme left to extreme right.

Inner sliders in the pan track make up the control bar. Each end of the control bar is pin jointed to a control slider. Cut-outs in the platform allow the control sliders to be placed underneath the platform. This eliminates interference between the moving sliders and the components on the top of the platform. Each control slider rotates each eye for coupled pan movement.

The pan stepper motor is mounted at the center of the plate with respect to the convergence mechanism via the stepper motor mounting plate. An absolute encoder is also mounted to the shaft of the stepper motor at the other side of the motor which does not drive the pinion.

Tilt Mechanism

The tilt mechanism consists of the tilt stepper motor, two universal joints, two L-bracket, two tilt bearings, a flex coupler, and two laser pointer mounts, as shown in Figure 21.

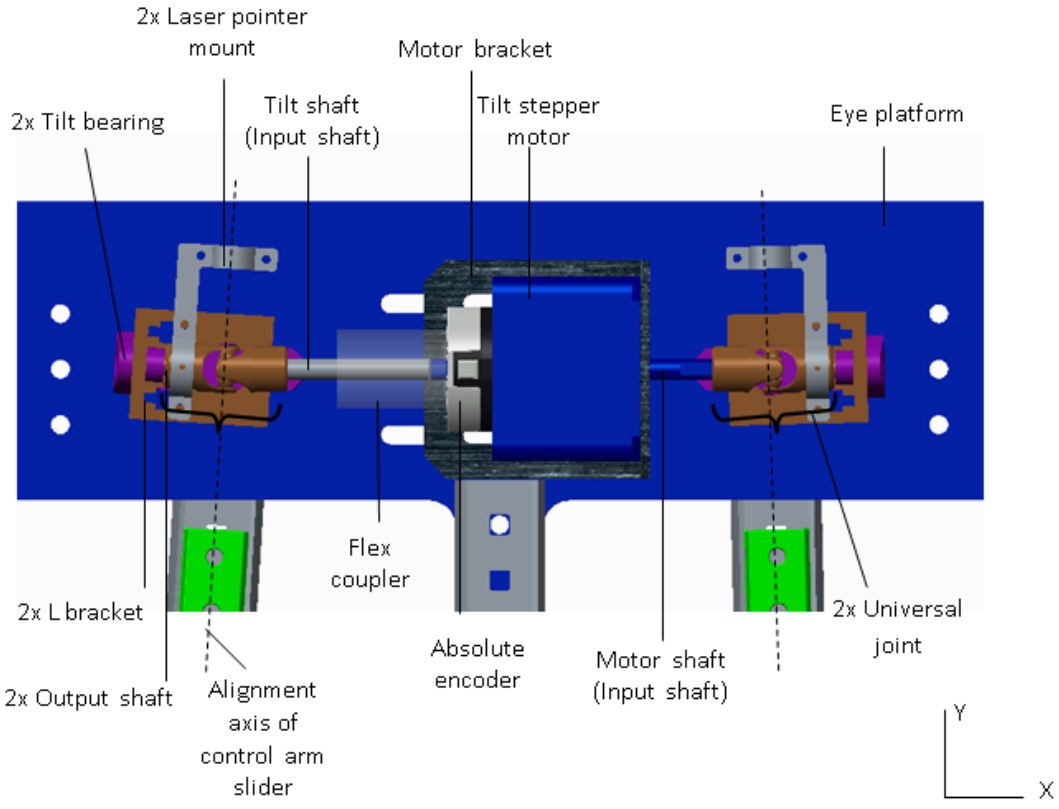


Figure 21: Top View of Eye L-Bracket and Tilt Mechanism

To conserve space and avoid gear backlash, the tilt mechanism is directly driven by a double-shaft stepper motor. The motor is positioned off-center between the two “eyes”. On the right side, the motor shaft is directly mounted to one end of a universal joint which drives the right “eye”; on the left side, a flex coupler will connect the motor shaft to a tilt shaft, which in turn connects to a universal joint which drives the left “eye”. If the motor were to be centered, two couplers would be needed to extend the motor shafts; the wide diameter of the standard-part flex couplers, however, caused interference problems with the rotating laser mounts, making this option unpractical.

While one side of each universal joint is connected to the tilt motor for tilt movement as described above, the other end of the universal joint contains “eyes” for pan and tilt movements. The shaft connected to the motor end of the universal joint for tilt movement is defined as the input shaft since it directly gives tilt inputs from the stepper motor. The shaft connected to the opposite end of the universal joint is defined as the output shaft since it is accepting tilt movements from the motor end of universal joint. Universal joints are critical in achieving independent pan and tilt movements. This coupling allows input and output shafts to rotate about their independently aligned axes. The output shaft, receiving pan inputs from the pan mechanism, changes its tilt rotation axis. By mounting eyes on the ends of universal joints which contain output shafts, simultaneous and independent pan and tilt eye movements can be achieved.

For our design, laser pointers represent “eyes” since laser pointers can visually show focus positions, which can provide useful information in testing. Laser pointers are mounted on the ends of the universal joint with output shafts via laser mounts. The detailed design of laser mount is discussed in Section 9.2.4. In order to achieve correct line of sight from panning, the central axis of the laser pointer ring which grips the laser pointer must always pass through the center of the universal joint and overlap with that of control slider (see Figure 22).

Figure 22 shows the detailed front view of tilt mechanism with L-bracket. The L-bracket transforms the pan movement from the control arm slider to the end of the universal joint with the laser pointer and the output shaft. The output shaft of each universal joint is connected to the L-bracket by a nylon flange (tilt) bearing. The L-bracket is rigidly connected to the control slider for pan movements by a Vex square shaft which is connected to a VEX lock plate (see Figure 23) mounted on the control arm slider. Therefore, the L-bracket can be rotated by the control arm slider without any radial slippage. Shaft collars are also mounted on the square shaft such that there is no axial translation of the L-bracket on the square shaft. The L-bracket is positioned on the fixed eye platform by using a pin-joint connection achieved with a VEX flat bearing (see Figure 23). To achieve correct line of sight from panning, the vertical central axis of the flat bearing and the VEX square shaft must pass through the center of the universal joint.

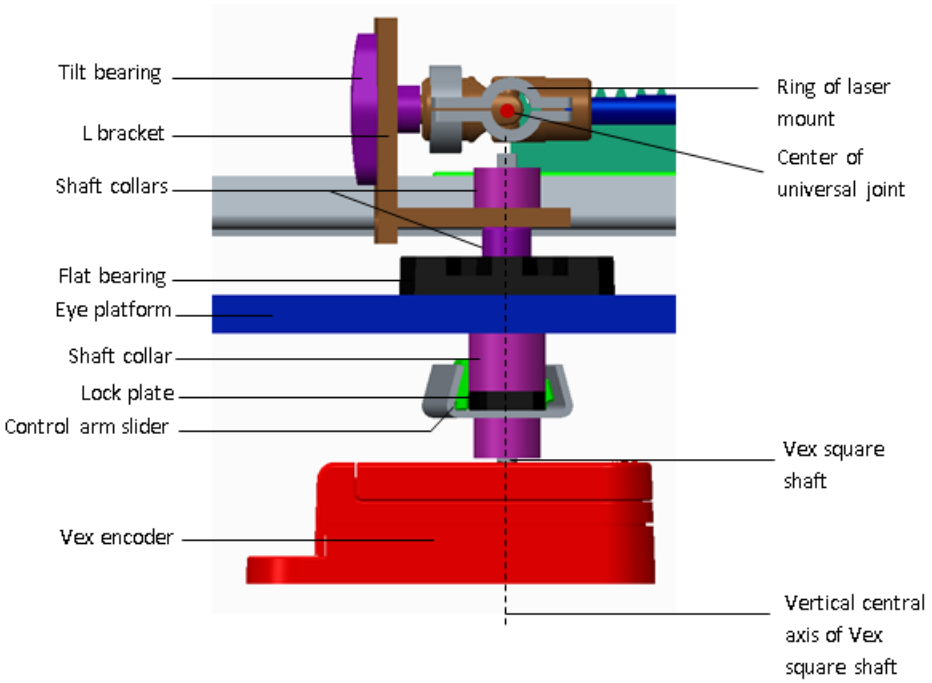


Figure 22: Detailed Front View of L-Bracket Assembly to the Eye Platform, Universal joint and Control Arm Slider

Figure 23: Vex Flat Bearing (Left) and Vex Lock Plate (Right) (Vex)

6.3.2 Final Design Parameters

Some of the initial input design parameters had to be adjusted and the output design parameters had to be optimized after the CAD model of a prototype was developed. The minimum distance between the control bar and the baseline cz_{min}, previously set to be 2 cm, had to be changed to 3.6 cm to accommodate the tilt motor. With that being changed, other output design parameters from Table 2 had to be modified. It was determined that it will be impractical to keep the rest of the input design parameters the same as those in Table 1 since output design parameters from Table 2 increase the workspace in length and width beyond 45 cm (18'') which is our design specification of the workspace.

Therefore, iterations were done in Mathcad (see Appendix C: Mathcad Final Design Parameters Calculation) using the same set of equations as explained Appendix A: Generating System of Equations to Determine Input and Output Design Parameters. However, input parameters in this Mathcad calculation are parameters that can be controlled and changed in the CAD model as listed in Table 3. The flexible input values are control bar length l_c and the maximum proximal distance of control bar from the baseline cz_{max}. Other parameters, such as cz_{min} and cx_{max}, have constraints on the workspace and the part geometry, as mentioned in Table 3, and they were set at the allowable extreme values which gave the most favorable design specifications.

The length of the control bar is coupled with critical design parameters such as minimum and maximum focal depth (fz_{min} and fz_{max} respectively) and range of vision ($2fx_{max}$). With the control bar length previously set as 13 cm, having cz_{min} increased to 3.6 cm caused fz_{min} to increase to 90 cm. Increasing the control bar length decreased minimum and maximum focal depth (fz_{min} and fz_{max} respectively) and range of vision ($2\ fx_{max}$), and therefore the length of the control bar was increased just until those design specifications ($fz_{min,}$ fz_{max}, and $2fz_{max}$) were close to our desired values. With the length of the control bar as 13.1 cm, the minimum focal depth and the range of vision R ($2fx_{max}$) achieved were 75 cm and 208 cm respectively. From this analysis of optimization, it was found that the length of the control bar was very sensitive to design parameters.

To still meet the design specification of maximum focal depth fz_{max}, the maximum proximal distance of the control bar from the baseline (fz_{max}) was increased to 13.1 cm by extending the length of convergence track. Consequently, the maximum focus depth was increased to 273 cm which exceeds our design specification of 250 cm.

Despite some of the design specifications not meeting initial design specifications, it was determined that the tradeoff is acceptable to preserve the workspace constraint of 45 x 45 x 45 cm^3. The final input and output design parameters are shown in Table 3 and Table 4.

Table 3: Final Input Design Parameters

Parameter	Description	Value	Justification
b	Baseline length	12.5 cm	Distance between two eyes-chosen on design specification
cz_{min}	Minimum proximal distance of control bar from the baseline corresponding to minimum focal depth	3.6 cm	The minimum allowance needed between the baseline and the control bar for no interference with the tilt motor
cz_{max}	Maximum proximal distance of control bar from the baseline corresponding to maximum focal depth	13.1 cm	Chosen to meet design specification of fz_{max}
r_x	Radius of pinion for rack x movement	1.905 cm	The standard radius of a Vex pinion
r_z	Radius of pinion for rack z movement	1.905 cm	The standard radius of a Vex pinion
ω_{pan}	Max angular velocity of focus point WRT center of baseline for minimum focal depth configuration	500 deg/sec	Saccadic human eye speed
$motor_{res}$	Number of steps in one revolution of the motor	200 step/ rev	Typical stepper motor resolution
cx_{max}	Max lateral distance of control bar to get the focal point from the center of the baseline to the farthest right/ left	5 cm	Chosen based on the maximum cut out that can be performed on Vex track and eye platform
l_c	Control bar length	13.1 cm	Chosen to optimize output design specifications

Table 4: Final Output Design Parameters

Parameter	Description	Value
W	Maximum dimension of the workspace in length	35.88 cm
l_s	Slider length	14.13 cm
fz_{min}	Max proximal distance of control bar from base corresponding to max focal depth	75 cm
fz_{max}	Max lateral distance of control bar to get the focal point from the center of the baseline to the farthest right/left	273 cm
Δfx	Pan resolution (smallest increment that focus point translates in x-direction)	1.25 cm
Δfz	Convergence resolution (smallest increment that focus point translates in z-direction)	1.25 cm
Δcx	Minimum proximal increment of control bar corresponding to x focal resolution Δfx	0.06 cm
Δcz	Minimum lateral increment of control bar corresponding to x focal resolution Δfz	0.06 cm
fx_{max}	Maximum distance in x-direction focus point can be panned from centerline (straight ahead).	104 cm
ω_{motor}	Max angular velocity of the pan stepper motor for control bar movement in x-direction	10 rev/s

7 Design Analysis

Throughout the design process before the development of a prototype, there were several iterations of design and analysis. The sections below discussed how theoretical test data points in a 3D space were calculated, how motors were selected to meet our design specification of pan speed and acceleration, and stress analysis of the eye platform.

7.1 Field of View and Focus Point Position Analysis

A left-handed Cartesian coordinate system with origin at the center of the baseline between the two eyes will be used to describe how the location of the focus point changes as each stepper motor changes the position of the mechanism. As shown in Figure 24: Field of View, the axes are oriented such that the positive x-axis is "right," the positive y-axis is "up" and the positive z-axis is "greater depth."

The range of vision in convergence is 0.75m at the smallest depth and 2.7m at the greatest depth. The eyes can tilt up to 45° above and below the horizontal plane for a total range of tilt of 90°. The range of vision in pan is 2.08m. Expressed differently, the eyes can pan up to 54.2° to the left or right of straight ahead at the minimum convergence depth and up to 21.1° to each side at the maximum convergence depth.

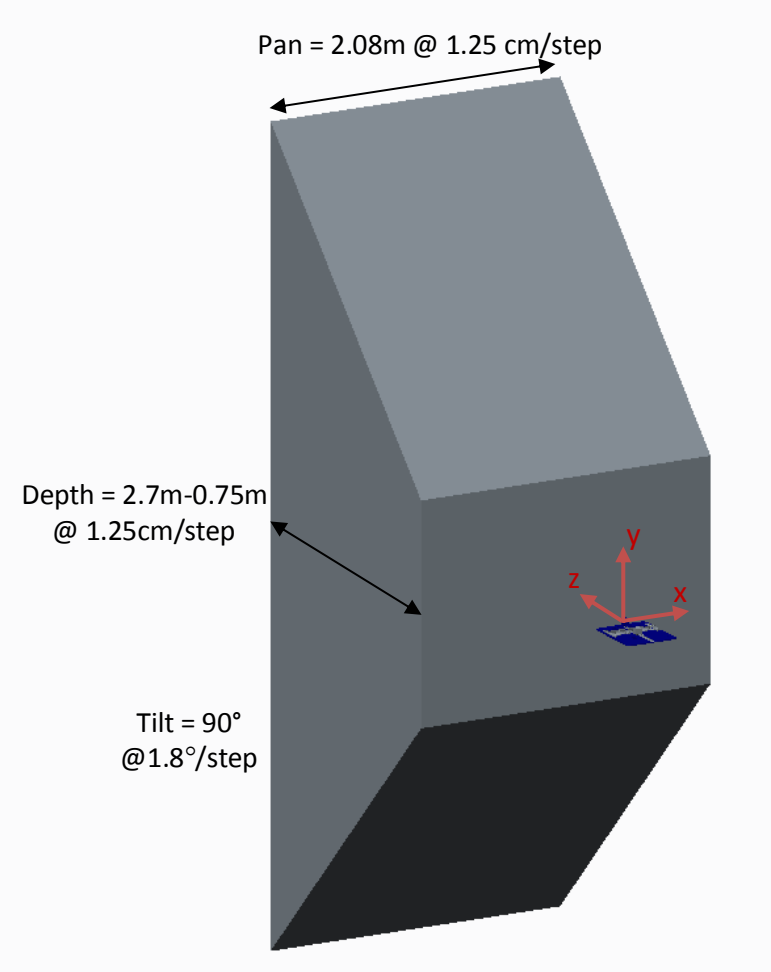


Figure 24: Field of View – All desired focus point positions are located in this space

Following is a description of how the focus point position changes as each stepper motor actuates the eyes.

When the pan stepper motor moves the control bar to the left (that is, in the negative x-direction) by an increment of ΔCx, the focus point moves to the right (in the positive x-direction) by an increment of ΔFx, where ΔFx is given by the following equation:

$$\Delta \mathrm{Fx} = \Delta \mathrm{Cx} \times \frac{\mathrm{Fz}}{\mathrm{Cz}}$$

As defined in Section 5.2, Fz is the focus point depth and Cz is the control bar depth, i.e. the distance of the control bar from the baseline. The focus point depth divided by the control bar depth is a constant ratio. For this design, with parameters given in Table 3 and Table 4, this ratio is equal to approximately 75 cm / 3.6 cm = 20.83.

Similarly, when the convergence stepper motor moves the control bar backwards (that is, in the negative z-direction) by an increment of ΔCz, the depth of the focus point increases (that is, it moves in the positive z-direction) by an increment of ΔFz, where ΔFz is given by the following equation:

$$\Delta \mathrm{Fz} = \Delta \mathrm{Cz} \times \frac{\mathrm{Fz}}{\mathrm{Cz}}$$

When the tilt motor rotates the tilt shaft to an angle θ, the focus point moves directly upwards or downwards (in the y-direction) to a height such that the angle between the horizontal (x-z) plane and the plane that bisects the focus point and the baseline is θ.

Note that while each increment ΔCx or ΔCz of the control bar always causes the focus point position to change by a constant distance (ΔFx or ΔFz)—regardless of the initial position of the mechanism or focus point—it is *not* true that each angular increment of the tilt shaft Δθ causes the focus point position to change by a constant amount (ΔFy). Instead, the tilt increment ΔFy depends on both the initial and final tilt angle ($\theta_{\mathrm{initial}}$ and θ_{final}), not just the difference in these angles. Thus:

$$\Delta \mathrm{Fy} = Fz \times (\tan \theta_{\mathrm{final}} - \tan \theta_{\mathrm{initial}})$$

It follows from the above relationship that the tilt resolution is best (i.e. ΔFy is the smallest) for small convergence depths and small tilt angles and worst for large convergence depths and large tilt angles (near 45° above and below the horizontal plane).

7.2 Torque and Speed Analysis

The pan motor had to provide enough torque to:

- overcome the friction in the sliders, and
- provide enough acceleration to reach our design specifications

At the specified torque output, the motor needed to be able to provide enough speed to fulfill our design specifications. The design specifications for the panning motion of the eye were 500 deg/sec angular velocity and 25,000 deg/sec^2 angular acceleration.

To determine the torque required to overcome frictional forces, the total mass being moved by the slider and the coefficient of static friction between Delrin and steel were taken into consideration.

To determine the rotational speed and torque required by the motor to reach the speed and acceleration design specifications of the eyes, speed and acceleration profiles for an eye from the left extreme to the right extreme were created. The profiles were then related to the control bar. In this way, the speeds and accelerations from the eyes were translated to corresponding speeds and accelerations for the control bar, and then the motor.

Determining the speed and torque requirements of the motor with this method of analysis would only be accurate if the eyes and control bar had similar speed and accelerations profiles. The velocity and acceleration profiles of the eyes shown in Figure 25 were created.

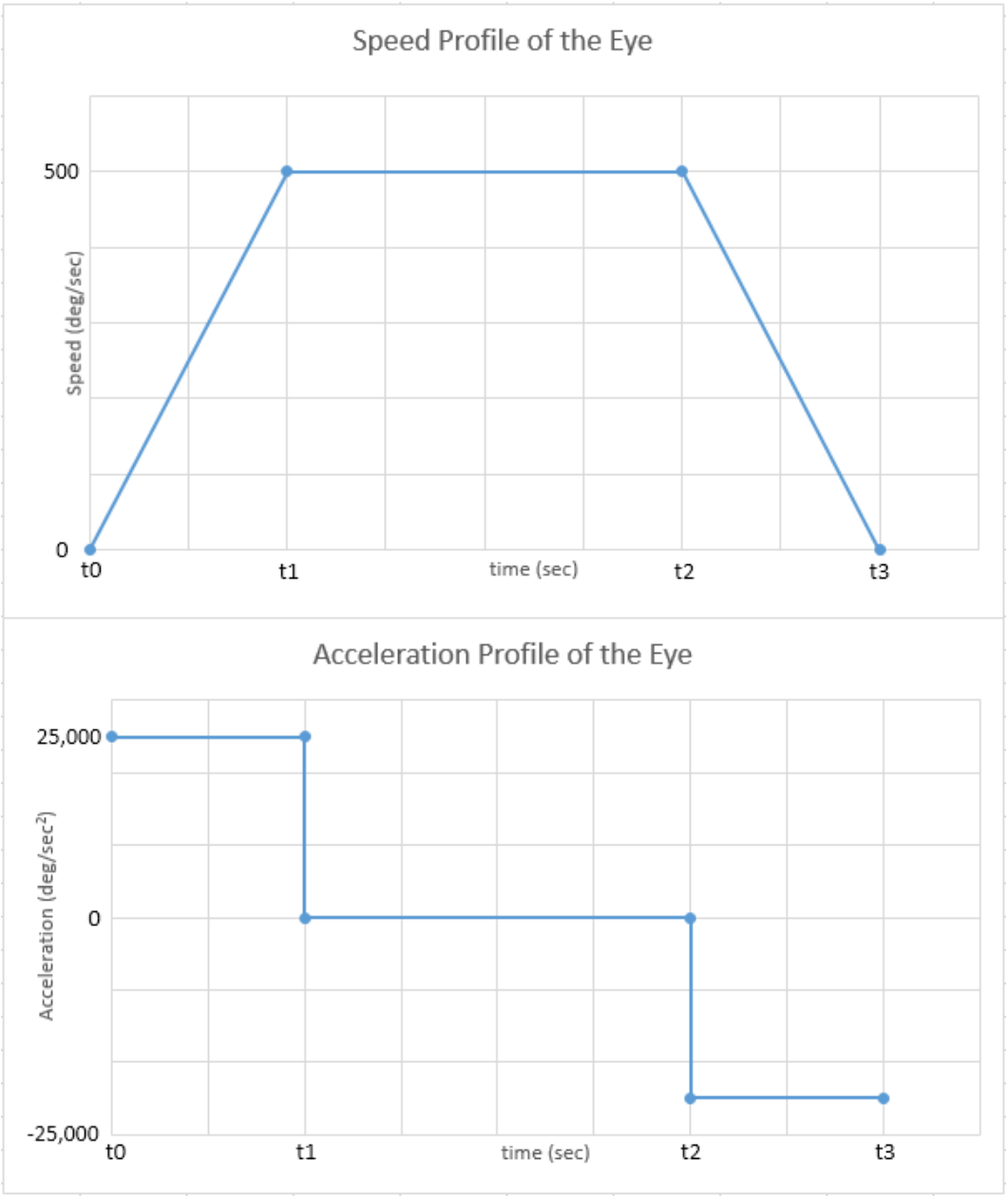


Figure 25: Speed and Acceleration Profiles of the Eye

Based on the speed and acceleration profiles in Figure 25, the following progression of accelerations was inputted into a kinematic mechanism analysis for one eye of the Creo model: 25,000 deg/sec^2 for 0.02 seconds, 0 deg/sec^2 for 0.06 seconds, and -25,000 deg/sec^2 for 0.02 seconds. The analysis yielded control bar speed and acceleration profiles in Figure 26 and Figure 27 similar to those of the inputted eye profiles. The control bar profiles are negative compared to the eye profiles because the direction of the rotational motion of the eye is opposite the linear direction of motion of the control bar.

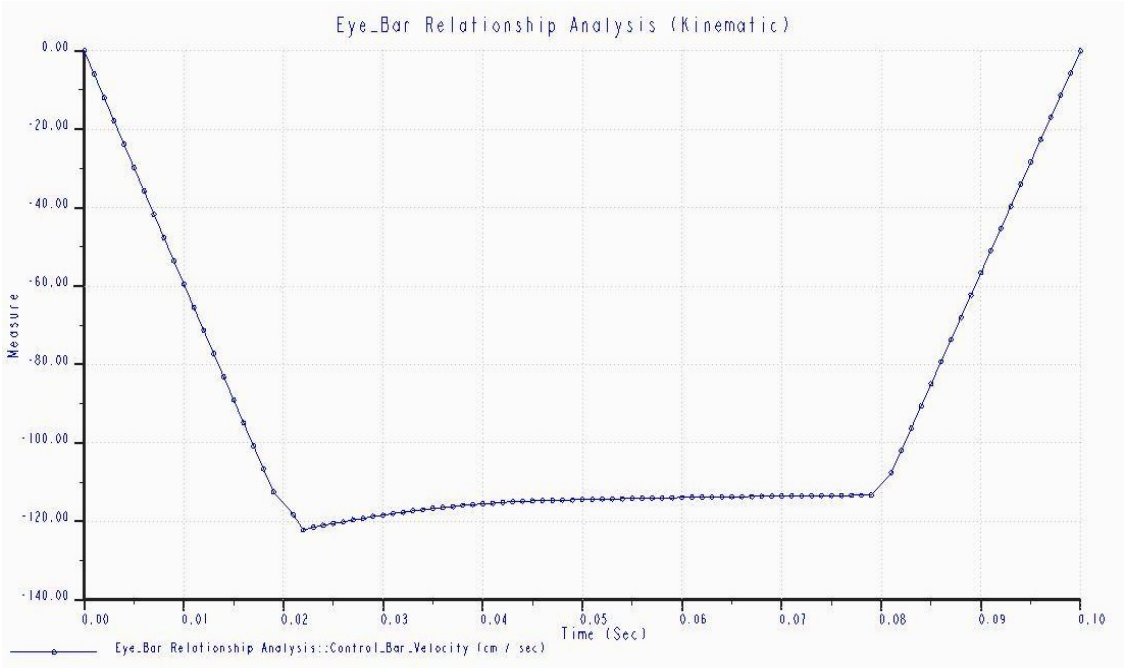


Figure 26: Creo-Generated Velocity Profile of the Control Bar

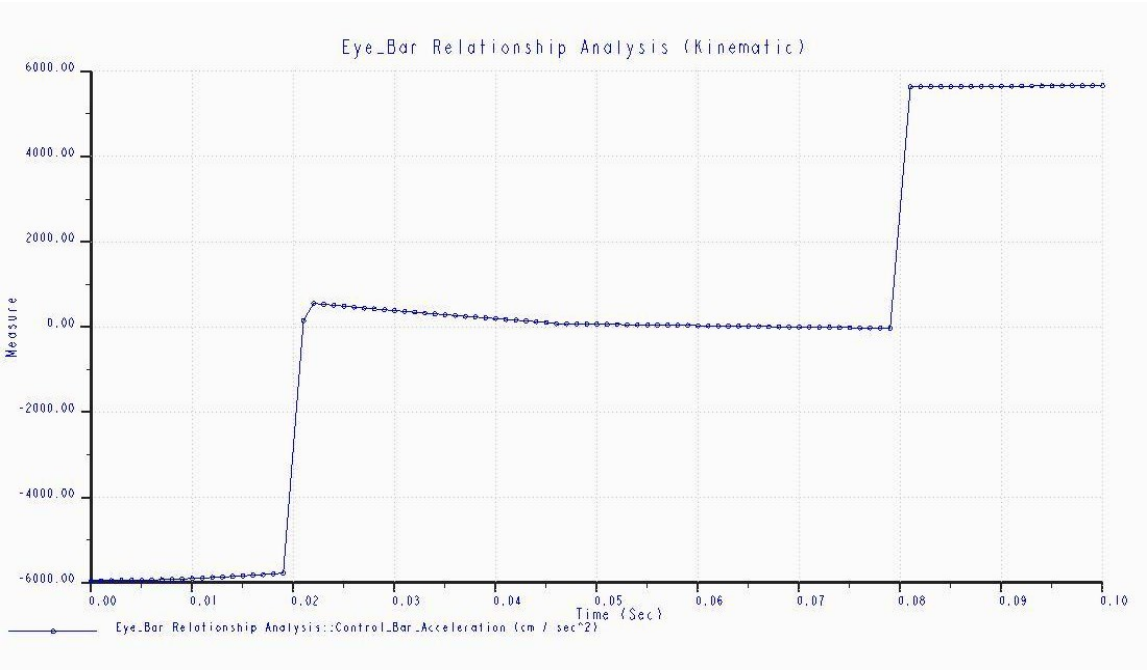


Figure 27: Creo-Generated Acceleration profiles of the Control Bar

Appendix D: Mathcad Torque and Speed Analysis contains the analysis for determining the necessary motor speed and acceleration specifications based on the similarities between the speed and acceleration profiles of the eyes and the control bar. Appendix D: Mathcad Torque and Speed Analysis also contains all of the assumptions and inputs for the model necessary for the calculations.

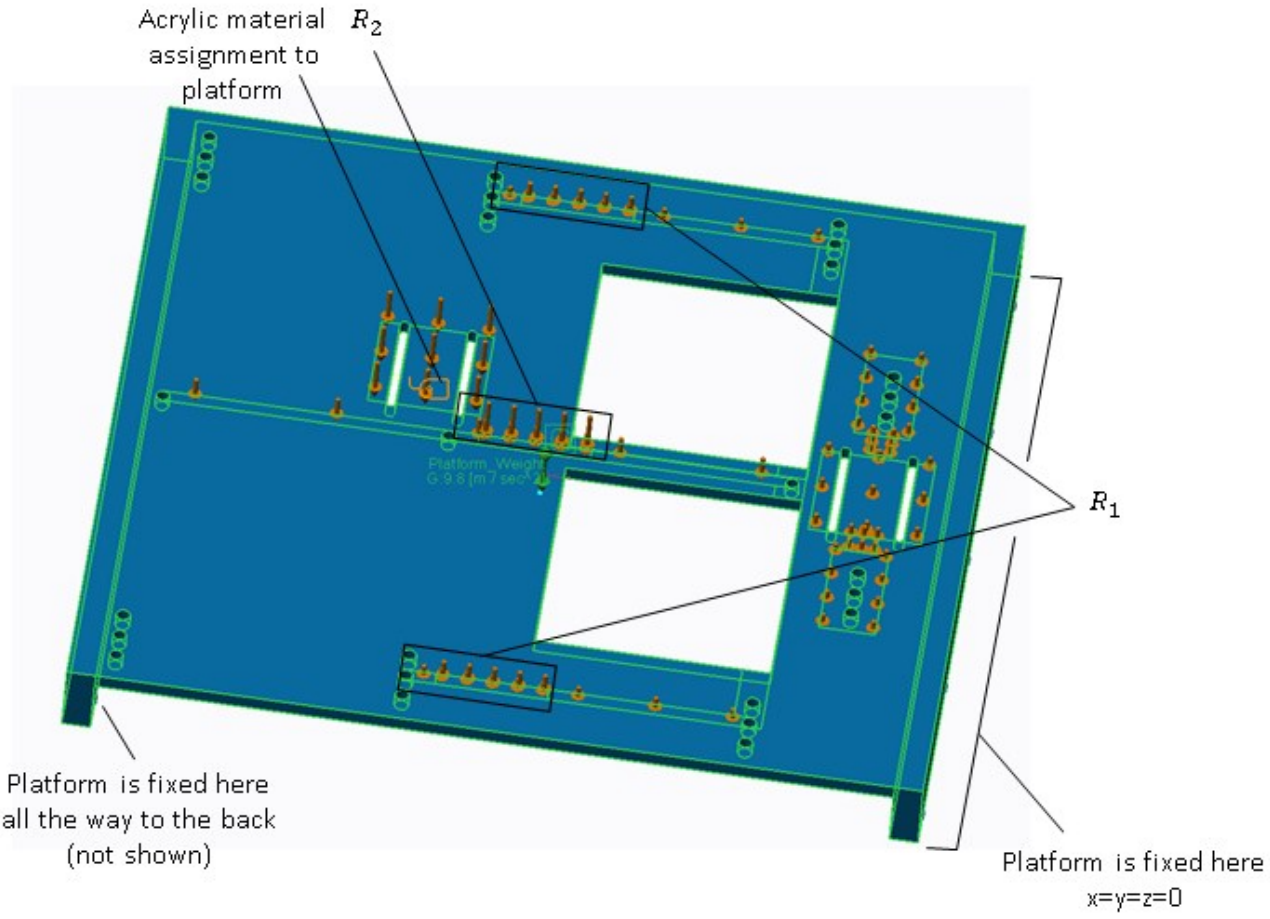
From the analysis, it was determined that a motor capable of a torque output of 31.7 Ncm and a speed output of 2089 pulses per second (pps) was necessary to achieve the design specifications for the speed and acceleration of the eyes. These specifications were specifically for the pan movements of the

eyes. It was decided that convergence or tilt speed and acceleration analyses were not necessary as convergence and tilt would not be as pronounced in the eyes as pan.

7.3 Stress Analysis

Because an initial evaluation of the base platform suggested the platform may be susceptible to bending due to the weight it supports, a stress analysis was performed to determine whether the platform would fail from bending stress by calculating deflections and safety factors. Finite element analysis was performed in Creo Parametric with the goal of visualizing the stress distribution of the plate so that the design of the platform configuration could be optimized before being manufactured.

A static stress analysis of the platform was performed at the early stage of design iteration. To perform a simple stress analysis, the platform was isolated from the assembly. The platform was put under fixed constraints as shown Figure 28. The material used for the platform is acrylic sheet with mechanical properties as shown in Figure 29(Physical Properties of Acrylic Sheets, 2013). The weight of the platform was considered and it was shown in Figure 28 by green highlights.



R_1, R_2= reaction forces from pan mechanism

Figure 28: Finite Element Model of Platform under Static Loads

Property[a] Mechanical		ASTM Method	Typical Value (0.236 Thickness)[b]	
	Specific Gravity	D 792	1.19	
	Tensile Strength	D 638	10,000 psi	(69 M Pa)
	Elongation		4.2%	
	Modulus Of Elasticity		400,000 psi	(2800 M Pa)
	Flexural Strength (Rupture)	D 790	16,500 psi	(114 M Pa)
	Modulus Of Elasticity		475,000 psi	(3300 M Pa)
	Compressive Strength (Yield)	D 695	18,000 psi	(124 M Pa)
	Modulus Of Elasticity		430,000 psi	(2960 M Pa)
	Shear Strength	D 732	9000 psi	(62 M Pa)
	Impact Strength		0.4 ft. lbs/in of notch	
	Izod Milled Notch	D 256	(21.6 J/m of notch)	
	Rockwell Hardness	D 785	M-94	
	Barcol Hardness	D 2583	49	
	Residual Shrinkage[c] (Internal Strain)	D 702	2%	

Figure 29: Acrylic Sheet Mechanical Properties (Physical Properties of Acrylic Sheets, 2013)

To determine the weight distribution from other parts in the platform, the materials information was added to the Creo assembly and the masses of parts were calculated. The platform was examined under static stress simulation using these masses as uniformly distributed loads over their corresponding coverage area on the platform (see Figure 28).

The mechanism which did not have direct contact on the platform is the pan mechanism as seen in Figure 30. The weight of the pan mechanism, W, was calculated by Creo. The weight of the stepper motor $W_{stepper}$ was acquired from the selected vendor, Schneider Electric Motion (see Section 9.1.2). To simplify the model, it was assumed that the pan mechanism without the stepper motor had a uniform weight distribution and that the weight of the stepper motor was only concentrated at the middle of the convergence track. Therefore,

$$\text{Reaction force at each of the two additional slider tracks} = R_1 = \frac{W - W_{stepper}}{3}$$

$$\text{Reaction force at the convergence track} = R_2 = W_{stepper} + \frac{W - W_{stepper}}{3}$$

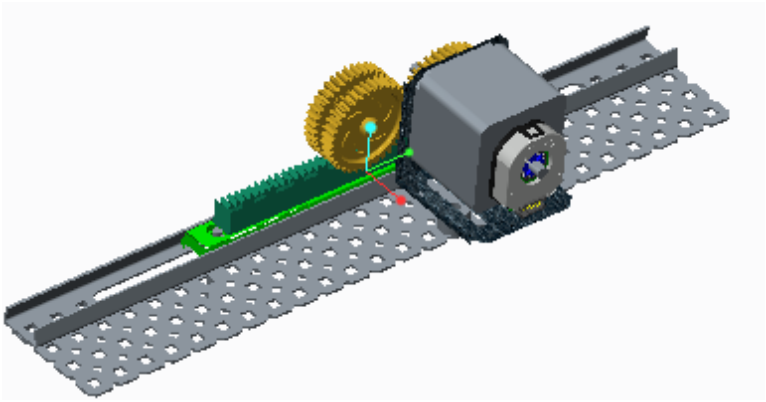


Figure 30: Pan Mechanism

Figure 31 and Figure 32 show the deflection and von Mises effective stress distribution after static stress analysis. In summary,

Maximum von Mises effective stress = 934 kPa

Maximum platform deflection in y direction = 0.038 cm

Tensile yield strength of acrylic sheet, at which a permanent deformation of the material occurs, is 73 times maximum value of von Mises effective stress. Therefore, the stresses and deflection were found to be negligible and the platform was safe against any static load. Both stress and deflection were maximum at the circled area close to convergence motor mounting slots and where the platform is cut open for the control pin to pass through for pan movements. This was the critical design area. Subsequent to this static stress analysis, fillets were added to the final design of the platform to reduce any concentrated stress.

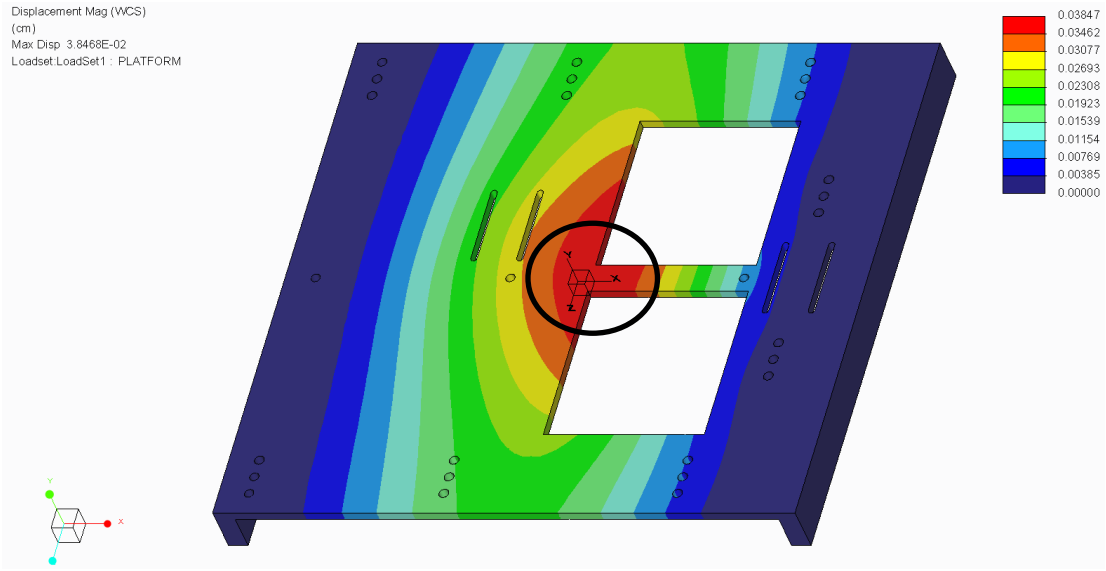


Figure 31: Platform y-deflection in cm

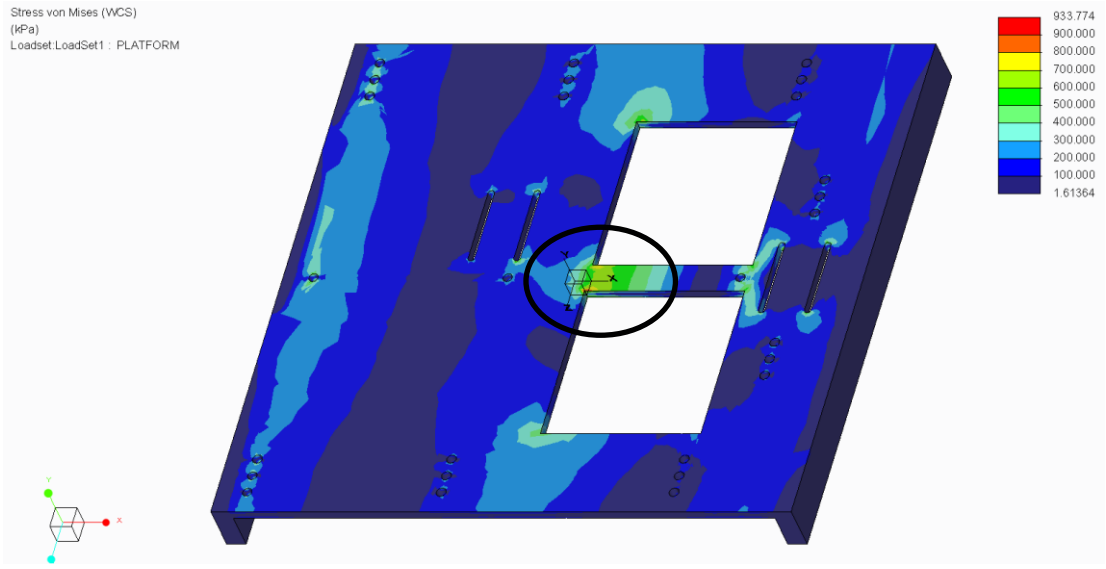


Figure 32: von Mises effective stress distribution in the platform

8 Control System Architecture Design

As depicted in Figure 33, the control system was designed to:

- drive and control actuators,
- monitor the system actuators' responses to system input, and
- monitor the output position, speed and acceleration of the platform eyes.

An embedded microcontroller (Arduino Uno) drives and controls three stepper motors. An absolute encoder is attached to each actuator to monitor their positions. An absolute encoder is also attached to each eye in order to monitor their angles. The encoders attached to the eye measure rotations about the y-axis, which is affected by pan and convergence motions. The encoder values of the tilt motor correspond directly with the tilt (rotation about the x-axis) of the eyes because the eyes are driven directly by the tilt actuator.

Each of the three actuators controls one of the three degrees of freedom of the platform eye pair: one for pan, one for convergence, and one for tilt. An embedded microcontroller determines the position changes of the actuators as they are driven by taking the difference between encoder readings, and then differentiates the position changes with respect to time to determine the actuator speed and acceleration.

Actuator motions are transformed to the eye motions through the platform pan, convergence and tilt mechanisms. The eyes experience orientation, angular velocity and angular acceleration changes due to the actuator outputs and the platform mechanical design. The encoder values are used by the processor to determine the rotational changes of each eye, which are then differentiated with respect to time in order to determine the angular velocity and angular acceleration experienced by each eye. The orientation of the eyes determines the position of the focal point.

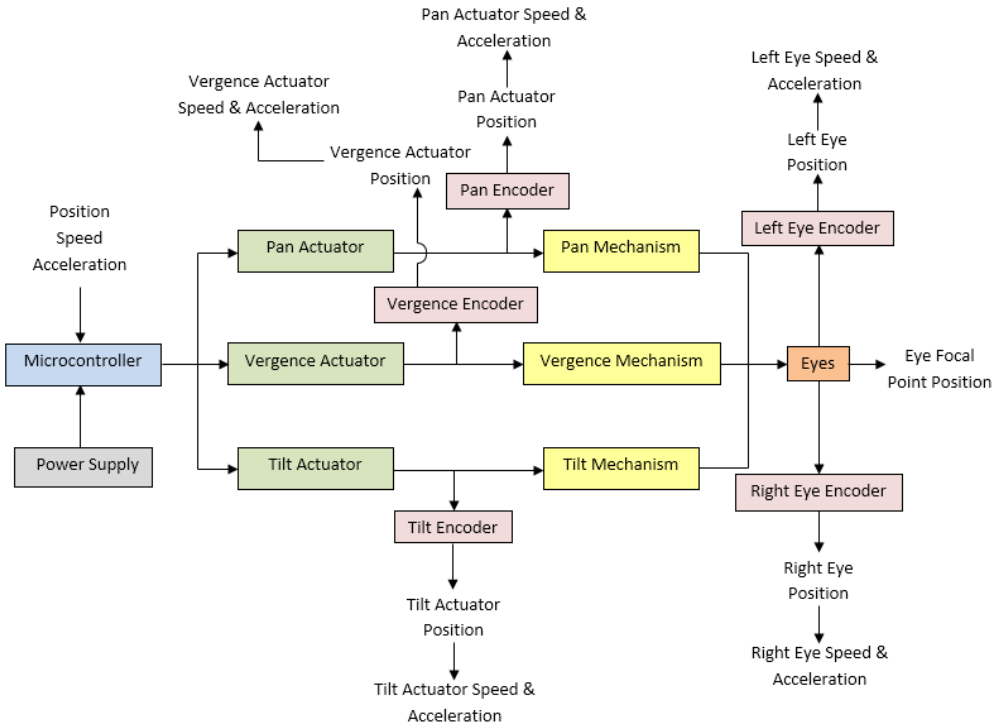


Figure 33: System Architecture Initial Design Flow Diagram

8.1 Mathematical Modeling of Pan Mechanism

In an effort to understand the kinetics of the mechanism to build an effective control system, the mathematical model of the pan mechanism was generated as the first step.

State space model is a mathematical representation of input, output and state variables related by first-order differential equations. A state space model was chosen for system modeling because it could easily identify inputs and outputs of a system in the vector space formed by the state variables. The control system design which is linked to the plant model determines input variables based on the set point (reference point) and the feedback it receives from output variables. The general representation of the state space model is

$$\dot{x} = Ax + Bu$$
$$\dot{y} = Cx + Du$$

where

x : state vector,

u : input vector,

y : output vector,

A : nxn state matrix (*n* is the number of states),

B : nxm state matrix (*m* is the number of inputs),

C : rxn state matrix (*r* is the number of outputs), and

D : rxm state matrix.

Bond graph modeling is an approach to build a state space model. Every bond comes with ṗ, f and q which refer to force (or torque), velocity (or moment) and displacement respectively in a mechanical system, and voltage, current and charges respectively in an electrical system. Each p and f of the bond graphs can be related to each other by a bond graph element such as I (mass moment of inertia), R or b (resistance or damping coefficient), C or k (capacitance or stiffness). The bonds are interlinked to each other by using 1 junction, 0 junction, GY and TF. Junction 1 means that the f parameter of all the bonds connected is constant and input force is equal to output force. On the other hand, junction 0 means that the p parameter of all the bonds connected is constant and input f is equal to output f. Junction GY is to connect a bond with input/output flow to another bond with output/input effort. Junction TF is to connect a bond with input/output flow to another bond with output/input flow.

This understanding of bond graph modeling was applied in modeling the panning motion as seen in Figure 34 and generating Equation (1) and Equation (2) in the following pages. In this bond graph, the model of the stepper motor used for rack *x* movement was left out since the Simulink Stepper Motor block will be used in the next step. Therefore, this bond graph model is purely mechanical. The torque given by the motor is transferred to angular velocity 1_{ω_x} in the pinion which in turn translates to $1_{v_{cx}}$in the rack. This translation velocity provides rotation in eye1 and eye2, $1_{\omega_{eye1}}$ and$1_{\omega_{eye2}}$ respectively.

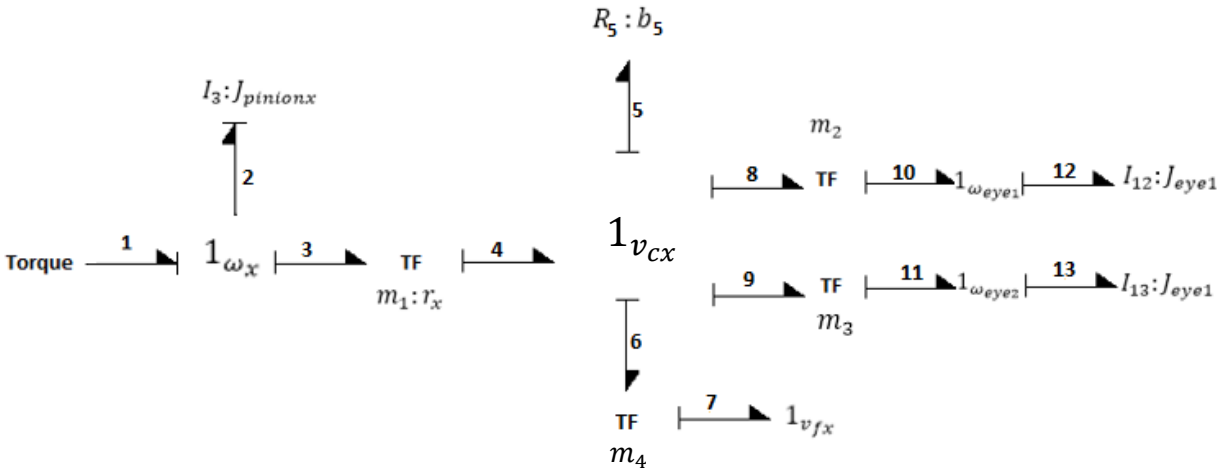


Figure 34: Bond Graph Model of Panning Motion

$$\dot{p_2} = \dot{p_1} - \frac{{r_x}^2 \times b_5 \times p_2}{J_{pinionx}} - {m_2}^2 \frac{{r_x}^2 \times J_{eye1} \times \dot{p_1}}{J_{pinionx}} - {m_3}^2 \frac{{r_x}^2 \times J_{eye2} \times r_x \times \dot{p_1}}{J_{pinionx}} \quad \textbf{(1)}$$

where $\dot{p_2}$= inertial torque of the pinion

$\dot{p_1}$ = torque applied by the stepper motor

r_x = radius of the pinion for the rack x movement

b_5 = damping ratio of the mechanical platform

p_2 =angular momentum of the pinion for the rack x movement

$J_{pinionx}$ = polar mass moment of inertia of the pinion

J_{eye1} = polar mass moment of inertia of eye 1

J_{eye1} = polar mass moment of inertia of eye 1

m_1 = r_x= ratio relating input and output flow velocities = $\frac{f_4}{f_3} = \frac{v_x}{\omega_x}$

m_2 = ratio relating input and output flow velocities = $\frac{f_4}{f_3} = \frac{\omega_{eye1}}{v_x}$

m_3 = ratio relating input and output flow velocities = $\frac{f_4}{f_3} = \frac{\omega_{eye2}}{v_x}$

m_4 = ratio relating input and output flow velocities = $\frac{f_7}{f_6} = \frac{v_{fx}}{v_{cx}} = \frac{fz}{cz}$

The ratios m_2 and m_3 are calculated from Figure 35 below.

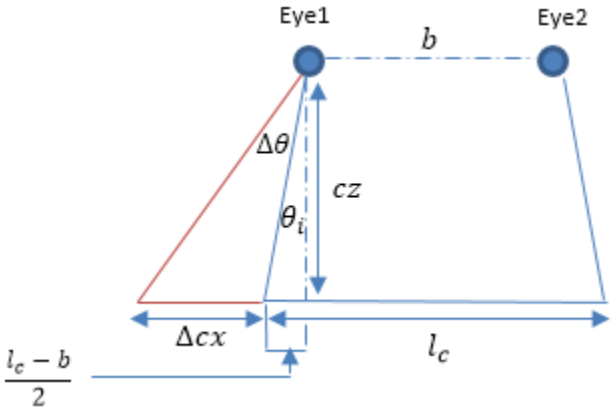


Figure 35: Control Bar Movement in x Direction which was Transformed into Eye Rotations

$$\tan(\theta_i + \Delta\theta) = \frac{\Delta cx + \frac{l_c - b}{2}}{cz}$$

By differentiating this with time, $$sec^2(\theta_i + \Delta\theta) \times \dot{\theta} = \frac{\dot{cx}}{cz}$$

$$\omega_{eye1} = \frac{v_x}{cz} \times cos^2(\theta_i + \Delta\theta)$$

$$\frac{\omega_{eye1}}{v_x} = \frac{cos^2(\theta_i + \Delta\theta)}{cz}$$

Similarly, $$\frac{\omega_{eye2}}{v_x} = \frac{cos^2(\theta_i + \Delta\theta)}{cz}$$

Since cz is a variable parameter in the model, this mathematical model becomes a linear time variant system. Tangent geometrical relation also introduces non-linearity to the system. Therefore, to simplify the control system, the worst case scenario could be assumed. To determine the worst case scenario, a simulation of the stepper motor control system needs to be performed in MATLAB Simulink by using a range of cz value from 3.6 cm to 13.1 cm (from design specifications mentioned in Section 6.3.2).

From Equation (1), it can be seen that the state variable is $\dot{p_2}$, relating to angular acceleration of the pinion. The input parameter is $\dot{p_1}$ which is the torque applied by the stepper motor. The output parameter is p_7 corresponding to the speed of the focal point and its ordinary differential equation is

$$p_7 = \frac{rx \times fz \times p_2}{cz \times J_{pinionx}} \quad (2)$$

Using Equation (1) and Equation (2), the state space model was generated. Since there is only one element in each of the vector column for all the variables, the model is reduced to a scalar representation with:

$\dot{x} = \dot{p_2}$ $\quad x = p_2$

$\dot{y} = p_1$ $\quad u = \dot{p_1}$

$$A = -\frac{{r_x}^2 \times b_5}{J_{pinionx}}$$

$$B = 1 - {m_2}^2 \frac{r_x \times J_{eye1} \times r_x}{J_{pinionx}} - {m_3}^2 \frac{r_x \times J_{eye2} \times r_x}{J_{pinionx}}$$

$$C = \frac{rx \times fz}{cz \times J_{pinionx}}$$

$$D = 0$$

8.2 Simulating System Plant Model in MATLAB

For the next step, simulation was performed in MATLAB Simulink using the existing stepper motor driver and stepper motor blocks from the Simulink library. It was determined that testing should be performed on the microcontroller alone rather than having it synced with the control system from MATLAB because Simulink can delay the data acquisition process. However, the simulation data obtained from MATLAB Simulink can be beneficial in effectively modeling and simulating the system response and implementing appropriate controls in a microcontroller.

The simulation period chosen in Simulink is 1 s. The parameters were chosen based on the selected Schneider Electric hybrid NEMA 17 stepper motor (see Section 9.1.2) with the selected Arduino Uno as the microcontroller (see Section 9.1.1) and an Adafruit stepper motor shield (see Section 9.1.3). Figure 36 shows the control system of the focal point by our pan stepper motor. The signal builder and stepper motor driver will give stepper motor inputs to the plant model. The plant model will generate output parameters, acceleration (a_fx), velocity (v_fx) and position (fx) of focus points which are to be controlled.

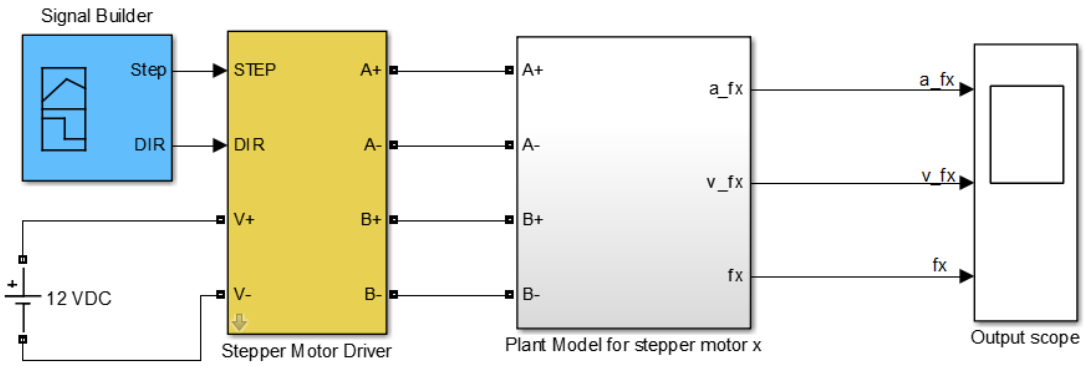


Figure 36: Control System in MATLAB Simulink

Signal Builder

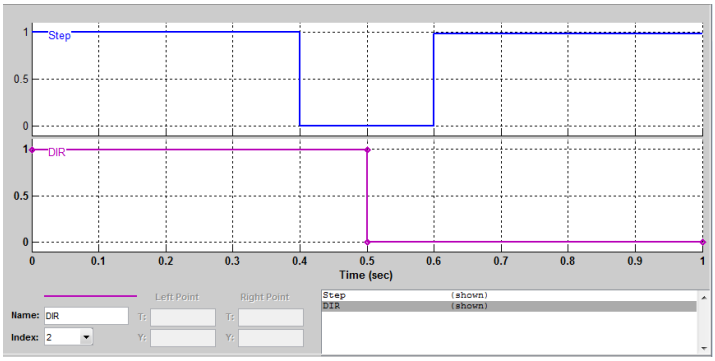


Figure 37: Screenshot of Step and DIR Signals Window

Step and Direction (DIR) signals are digital inputs. In this simulation, step signal (Step=1) tells the motor to rotate in the positive direction (DIR = 1) with the specified speed in stepper motor driver for 0.4 s, stop (Step = 0) for 0.2 s and then rotate (Step = 1) in the negative direction (DIR = 0) with the specified speed in the stepper motor driver.

Stepper Motor Driver

Simulink has a stepper motor driver block set up. Phase current was set to be 1.5 A which was obtained from our stepper motor specs. Stepping rate was set to be 1671 step/s, the speed that was needed to overcome the torque and achieve a focal point acceleration of 25000 deg/s^2 as calculated in Section 7.2. Sampling rate was set to be 1ms to provide reasonable accuracy for a high Pulse Width Modification (PWM) an Arduino Uno may receive.

Plant Model for Stepper Motor

The plant model masks a hybrid stepper motor block which is linked with blocks to calculate dynamics and kinematics of the focal point. (a_fx, v_fx, fx) (see Figure 38).

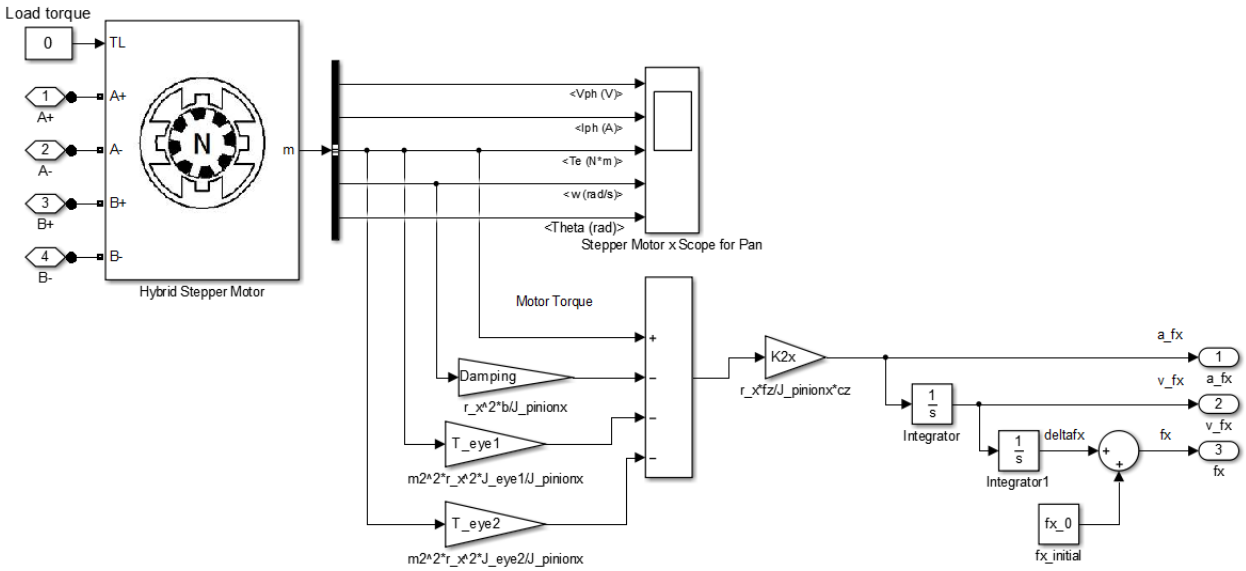


Figure 38: Plant Model for Stepper Motor

The parameters chosen for the hybrid stepper motor block as seen in Table 5 were taken from the specification sheet provided by Schneider Electric stepper motor. Maximum Flux Linkage and total friction of the motor are unknown at this point. The former can be obtained experimentally by driving the motor to a constant speed N (rpm) and by measuring the maximum open-circuit winding voltage Em (V). The parameter ψm is then computed by the following relation:

$$\psi m = (30/p\pi)(Em/N),$$

where p is the number of pole pairs given by p = 360 / (2m·step). Here m = phase number, step =step angle in degrees (MATLAB Simulink Blocks: Stepper Motor, 2014).

Table 5: Parameters for Hybrid Stepper Motor Block

Motor type	Permanent-magnet/ Hybrid
Number of phases	2
Winding Inductance	5e-3 (H)
Winding Resistance	2.1 (Ohm)
Step angle	1.8 (deg)
Maximum Flux Linkage	
Maximum Detent Torque	0.015 (N.m
Total Inertia	57e-7 (kg.m.m)
Total friction	
Initial speed	0 (rad/s)

The stepper motor block receives voltage inputs from the stepper motor driver block and then generates five signals: electromagnetic voltage, phase current, motor torque, rotor speed and rotor position. Using motor torque and speed, Equation (1) and Equation (2) from Section 8.1 can be numerically simulated in Simulink to calculate angular acceleration which was successively integrated to obtain speed and position of the focus point.

Due to time constraints, parameters such as damping coefficient b, polar moments of inertia of eye1, eye2 and the pinion, maximum flux linkage and inertia of the motor were not determined. As a result, the full simulation could not be performed in MATLAB, but the presented mathematical model will be useful for future implementations of advanced control for our eye mechanism.

9 Prototype Development

The mechanism was designed such that commonly available standard vendor parts could be used and only minor machining and manufacturing of parts was required. The first section of this chapter discusses which mechanical and electromechanical parts were required and how standard vendor parts were evaluated before purchase. The second section discusses how the custom parts were manufactured and modified. The last two sections discuss the assembly of the prototype and the implemented control system architecture.

9.1 Selection of Purchased Parts

From the control system architecture, mechanical design and analysis, it was determined what mechanical and electromechanical components should be acquired. The critical components for the development of our electromechanical system are listed below.

9.1.1 Arduino Uno

Arduino Uno is the microcontroller board for giving inputs to stepper motors and acquiring information from sensors. This microcontroller comes with an integrated development environment (IDE) which runs programs, written in C or C++, on personal laptops. This microcontroller was selected because it is inexpensive, readily available, and compatible with the selected stepper motor shields (see Section 9.1.3 Stepper Motor Shield).

9.1.2 Stepper Motor

Different types of actuation were reviewed to drive the mechanism based on design specifications for velocity, acceleration and resolution. Stepper motors were chosen because of their high torque capacity relative to their compact size. Unlike DC motors and servo motors, stepper motors rotate in increments which give them high accuracy. Depending on the type of stepper motor operation mode, each stepper motor has a defined number of steps for each revolution.

To decide on a particular stepper motor that would meet the torque and speed requirements specified in Section 7.2, various stepper motors' torque-speed graphs were reviewed. A torque-speed graph gives information on the maximum torque capacity of a stepper motor at a particular speed. Generally, the motor torque decreases as the speed of the stepper motor increases.

Assuming an eye pan angular velocity of 500 deg/s and angular acceleration of 25,000 deg/s, a mass of 0.2543 kg being moved by the pan mechanism, and a pan pinion radius of 1.905cm, the analysis in "Appendix D: Mathcad Torque and Speed Analysis" yielded a pan motor requirement of maximum torque and rotational speed of 31.7 N-cm and 2089 pulses per second (pps) respectively. There were no specifications for velocity and acceleration of convergence and tilt because they did not have high impacts on pan parameters which were important for the design concept. However, the motor for convergence movements had to be as powerful as that for pan movements because the convergence motor was required to carry the weight of the pan motor itself in addition to the loads supported by the pan motor. It also needed to overcome the static friction in the convergence mechanism. On the other hand, the tilt motor, which directly drives the eyes, only required a small amount of torque and therefore, a less powerful motor was used.

After studying product specifications of various stepper motors as well as stepper motor shields to drive the stepper motors, three NEMA 17 1.8° (200 steps per revolution) 2-phase hybrid stepper motors were chosen from Schneider Electric Motion(see Figure 39). The two stepper motors used for

pan and convergence movements each had a 23 N-cm holding torque capacity. Figure 40 shows that at approximately 2000 full steps per second (2000 pps), the stepper motor has a torque capacity of 22 N-cm at 24 VDC and 32.5 N-cm at 48 VDC. If the latter voltage were chosen, the pan specification for velocity and acceleration could be met. However, the stepper motor shield for driving the stepper motor had an input voltage limit of 12 V (see Section 9.1.3 Stepper Motor Shield). An input voltage of 48 V to the stepper motor would require a different type of micro-controller and electrical circuit design beyond the scope of this mechanical engineering project. Therefore, it was expected that some of the velocity and acceleration specifications would not be met during testing. The stepper motor for tilt movement had a smaller torque capacity of 22.6 N-cm. Its torque-speed graph is shown in Figure 41.

Figure 39: Purchased Stepper Motor from Schneider Electric Motion (Schneider Electric Motion, USA)

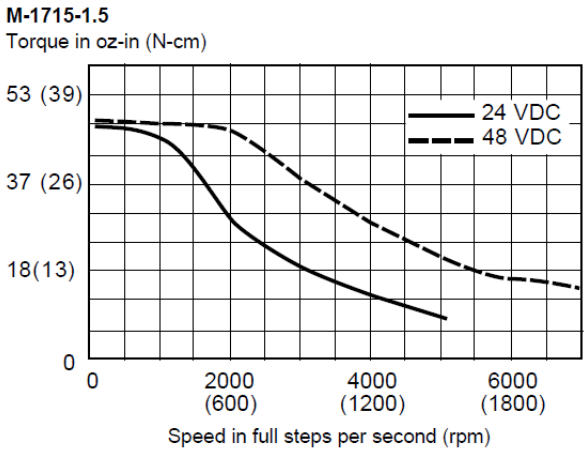


Figure 40: Torque-Speed Graph of Stepper Motor Used for Pan and Convergence Movements (Schneider Electric Motion, USA)

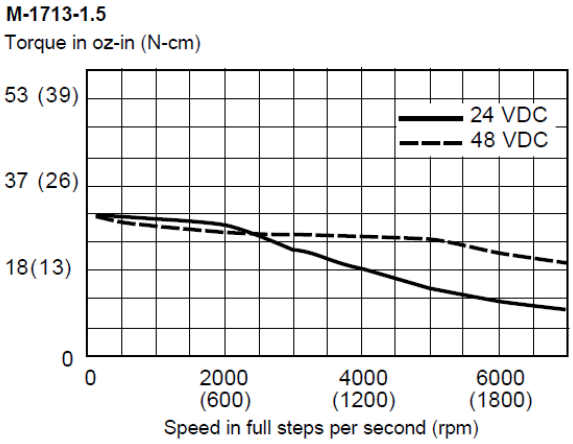


Figure 41: Torque-Speed Graph of Stepper Motor Used for Tilt Movement (Schneider Electric Motion, USA)

These three stepper motors are bi-polar type with four wires. Four wires, upon receiving the current from the power supply, create inductance in four windings and polarize the corresponding electromagnets with geared teeth. To rotate the geared rotor surrounded by four of these electromagnets, pulses need to be supplied to windings in sequential order. Unlike unipolar stepper motors which only use half of each winding, bipolar motors fully utilize the whole winding to create inductance and hence, they possess higher torque capability.

In addition to superiority in torque performances, these stepper motors were also chosen because of their adaptability to work in the designed mechanism. The double shaft option allows the stepper motor to be used in the designed tilt mechanism without modification. The stepper motors also come with mounting holes for mounting encoders. Both the mounting hole pattern and the shaft size of 5 mm are compatible with absolute encoders that were purchased.

9.1.3 Stepper Motor Shield

As explained above, stepper motors rotate by creating inductance in the coil to polarize the electromagnet wound inside. Inductance is related to voltage pulses by

$$V = L\frac{di}{dt}$$

For a pulse voltage supplied by a micro controller, the rate of change of phase current results in inductance. If the stepper motors are directly wired to digital output pins of a microcontroller, the microcontroller may occasionally instantaneously draw too much current and be greatly damaged. In addition, voltage pulses have to be turned on/off to successively create inductance in motor windings so that the rotor can rotate continuously. From the above equation, this creates rapid change in currents, causing voltage spikes which the micro-controller cannot handle. Therefore, to accommodate this special operation of a stepper motor, a motor power supply separate from Arduino Uno is ideal.

To eliminate the need for electrical circuit design, an existing circuit board was purchased. Although there are a number of stepper motor drivers and circuits, they need modification and understanding of electrical circuits to work with the acquired microcontroller, Arduino Uno. A Motor Shield from Adafruit was selected because it is capable of running the stepper motor using an Arduino Uno microcontroller while addressing all of the current, voltage and inductance problems mentioned above at a low cost. The shield has two ports for stepper motors (either unipolar or bipolar). It is a stackable design with 5 address select pins to be able to control up to 64 stepper motors. For this

application, three stepper motors needed to be controlled and therefore two Adafruit motor shields were purchased.

9.1.4 Encoders

Encoders give information on mechanical motion such as speed, direction and shaft angle due to changes in electrical signals. Among the different shapes and sizes of encoders, rotary encoders were considered for monitoring both:

- the position of the stepper motors in order to verify that the mechanism has gone through the desired actuation, and
- the angular orientation of the two eyes in order to use the changes in orientation to validate the design specifications for eye angular velocity and acceleration.

Inside the encoder, there is a disc to which two channels are connected. When the shaft rotates, one channel blocks and passes the signal alternately and vice versa for the other shaft. Consequently, one channel from the encoder gives digital outputs of “HIGH” while the other channel gives digital outputs of “LOW”. There are essentially two types of encoders – absolute and incremental. Incremental encoders give electrical signals in a chain of square waves whereas absolute encoders give a distinguished digital word for each position of the encoder as capable by its stated resolution. In other words, absolute encoders tell the exact rotary position unlike incremental encoders which, can only measure relative mechanical position with respect to the initial position.

9.1.5 Absolute Encoder

To keep the kinematic equations in Appendix A: Generating System of Equations to Determine Input and Output Design Parameters valid, the mechanism needed to be calibrated to initial input parameters. Absolute encoders, recording the absolute mechanical position in digital values, retain their values even after power is cut off. This makes them very useful for calibration purposes because once calibration is performed to relate encoder digital values to the actual mechanical position, the configuration of the mechanism can be calculated if other dimensions of mechanical parts used in the theoretical kinematic equations are constant. Therefore, absolute encoders were chosen over incremental encoders to calibrate stepper motors to the initial mechanical configuration.

The acquired absolute encoders were of the model CUI AMT203-V (see Figure 42). It is a capacitive encoder with both absolute and incremental capabilities and a resolution of 1024 pulses per revolution. This resolution is higher than the stepper motor resolution of 200 steps per revolution and therefore, these encoders were more than adequate. In addition to favorable electrical characteristics, the mechanical chassis of these encoders can be mounted on the hole pattern included on the stepper motor. They have an open center and come with hub adapters varying between 2 mm and 8 mm diameters. The hub adapter of 5 mm diameter can be easily driven by the stepper motors with a shaft diameter of 5 mm.

Figure 42: CUI Absolute Encoder (CUI Inc.)

9.1.6 Vex Optical Shaft Encoder

Vex optical shaft encoders were chosen to record the eye orientation (see Figure 43). It is an optical incremental encoder and has a resolution of 90 pulses per revolution. Despite its low resolution, it was still favorable due to its square opening which is compatible with the Vex square shafts attached to the eyes. Moreover, since the main purpose was to record the eye angular velocity and acceleration rather than to be used for sending feedback needed for a control system, the specifications of vex encoders were sufficient.

Figure 43: Vex Optical Shaft Encoder (Vex Robotics)

9.1.7 Tilt Bearing

A bearing was required to mount the tilt shaft on the L bracket. The bearing chosen was a nylon flange-mounted sleeve bearing that fit the general size requirements (see Figure 44).

Figure 44: Nylon Flange Mounted Sleeve Bearing (McMaster-Carr)

9.1.8 Universal joint

The specification for maximum allowable shaft misalignment was important for selecting the appropriate universal joint. Speed and torque specifications were less prioritized because a small load was supported and there was no requirement for tilt velocity. The universal joint chosen was a ball and socket plastic joint from McMaster-Carr with a maximum shaft misalignment of 45°. This provided limitation on range of vision at the minimum focus depth because to be able to position focus point at 104 cm left and right from the origin, a maximum shaft misalignment of 56° was required. When actual testing was performed, it was found that the universal joint was able to rotate more than 45°, but this could be a source of error as discussed in Section 10.1.4.

The selected universal joint was also designed to be mounted easily on the shaft. By having the tilt shaft drilled with through holes of the same size as those included in the U-joint, spring pins could be inserted to connect the joint to the tilt shaft. The inner diameter of the universal joint is also the same as that of the tilt shaft. Lastly, this U-joint is compact and relatively cheap.

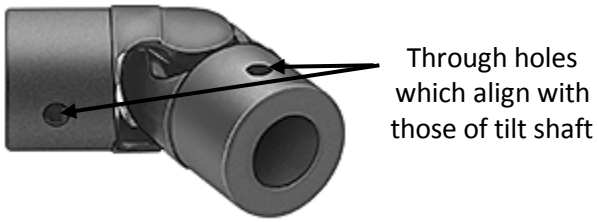


Figure 45: Ball and Socket Plastic U-joint (McMaster-Carr)

9.1.9 Laser Pointer

The focus point position needed to be measured to verify the design specification. To measure the focus point position, two laser pointers were mounted to represent “eyes”. The chosen red laser module was small enough to be mounted on the universal joint, and also powerful enough to project the precise laser point at a large distance with 5V input voltage port from the Arduino Uno.

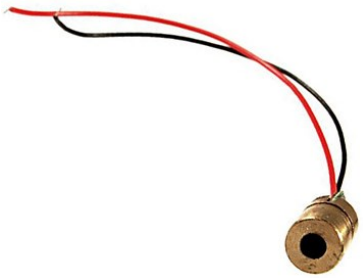


Figure 46: Red Laser Module (MiniInTheBox)

Table 6 lists all mechanical and electromechanical components that were purchased and used in the final prototype.

Table 6: List of Purchased Standard Vendor Parts

Name	Description	Quantity	Unit Price	Total Cost	Vendor	Part Number
Universal Joint	Plastic, 3/16" shaft diameter, max operating angle 45°	2	$17.05	$34.10	McMaster-Carr60625K92	60625K92
Shaft for tilt	3/16'' diameter, 6'' length Cut into 2x 1.5 cm 1x 4 cm	1	$4.79	$4.79	McMaster-Carr	1327K52
Flex coupler for tilt motor	5mm to 5mm shaft	2	$4.95	$9.90	Adafruit	1175
Stepper motor shield for Arduino	4 H bridges (TB6612 MOSFET driver), PWM driver chip, Up to 2 servos, Up to 4 bi-directional DC stepper motors, Up to 2 stepper motors (unipolar or bipolar)	2	$19.95	$39.90	Adafruit	1438
Laser pointer	Red; cylindrical casing; 6 mm OD, 3.5-4.5 V	2	$1.99	$3.98	MiniInTheBox.com	#00077236
Bearing (for tilt shaft)	ID = 3/16” (or 5mm) to accommodate tilt shaft	2	$3.20	$6.40	McMaster-Carr	2827N2
Stepper motor (for tilt)	Double shaft, NEMA17, hybrid, holding torque 32 oz-in, 0.3kg, 1.5 A phase current	1	$55.51	$$55.51	Schneider Electric Motion	#0119
Stepper motor (for pan, convergence)	Double shaft, NEMA 17, hybrid, holding torque 60 oz-inch,0.347 kg, 1.5 A phase current	2	$59.57	$119.14	Schneider Electric Motion	OMHT17-275-D
Motor mounting bracket (tilt)	NEMA 17 size	3	$8.95	$26.85	Adafruit	1297
Absolute encoder	Absolute encoder, 5mm open center	3	$48.65	$145.95	DigiKey (supplier)	102-2050-ND
Linear Motion Kit	Vex 2x 12" track 2x 17.5" track 4x Delrin inner slider 4x Delrin outer slider 2x steel bracket	1	$24.99	$24.99	Robot Mesh	SKU: 276-1926
Rack Gear (16-pack)	16 Vex rack gears (one available size)	1	$19.99	$19.99	Robot Mesh	SKU: 276-1957
Optical Shaft Encoder	Vex Quadrature encoder (2 pack)	1	$19.99	$19.99	Robot Mesh	SKU: 276-2156
Total				$516.87		

9.2 Design of Manufactured Parts

The mechanism's uniqueness meant that the exclusive use of standard parts could not suffice for prototyping the design. Thus, manufacturing of new parts and modification of standard parts was necessary. Several different methods were employed to manufacture these parts. Drawings of the manufactured parts can be found in Appendix I: CAD Drawings of Manufactured Parts.

9.2.1 Acrylic Base and Mounting Plate

The CAD model of the platform base, shown in Figure 47, used in the CAD assembly was finalized to include all of the necessary cutouts (to allow the control bar pin joints to pass through the base and move without interference) and through holes for mounting components. The final base design measured 14" by 17" and was laser cut from ¼" thick acrylic sheet.

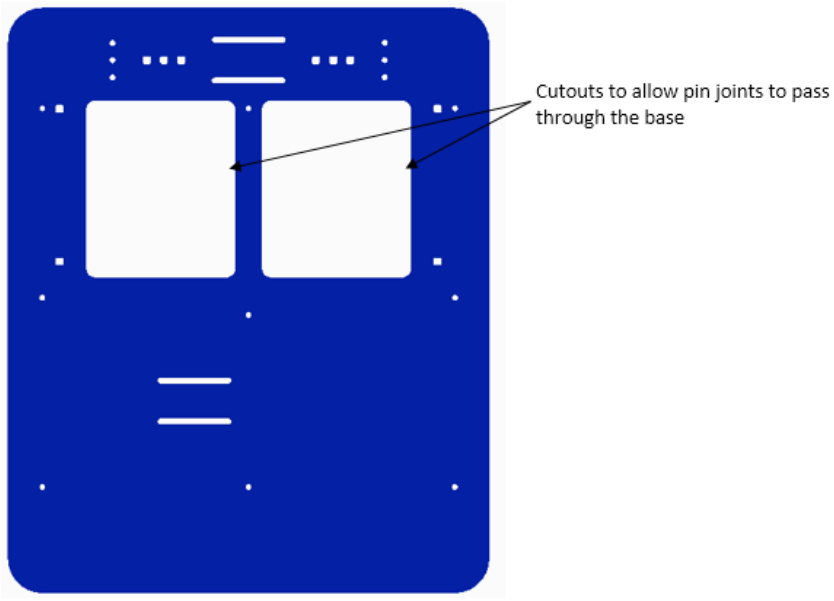


Figure 47: CAD Model of Platform Base

Additionally, the mounting plate used to mount the pan mechanism to the convergence slider was laser cut. In the CAD assembly shown in Figure 20, the mounting plate is pictured as a VEX plate; however, a VEX plate of the required length was not readily available. For convenience, a simplified plate of the desired length was modeled with only the required holes needed for mounting, as shown in Figure 48, and was laser cut from ¼" acrylic.

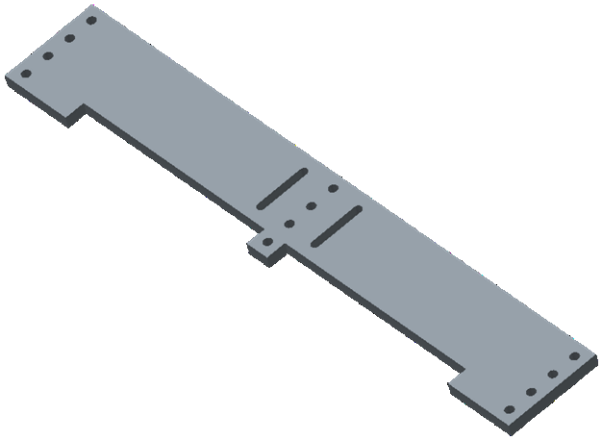


Figure 48: CAD Model of Acrylic Mounting Plate

9.2.2 Custom Control Bar Sliders

In the design, the control bar consisted of three inner slider pieces. A standard VEX slider is made of Delrin plastic and has four 8-32 tap drill holes spaced along its length. In order for the control bar to be the desired length of 13.1 cm, as determined from numerical analysis in Mathcad, the spacing of these holes had to be changed. It was found from the Mathcad program that output values for range of vision are sensitive to the control bar length, and the link length that would result from the standard VEX slider hole spacing would not be adequate. The custom slider was therefore created in Creo (see Figure 49) and was printed on the Dimension SST 1200ES Rapid Prototype Machine with ABS plastic.

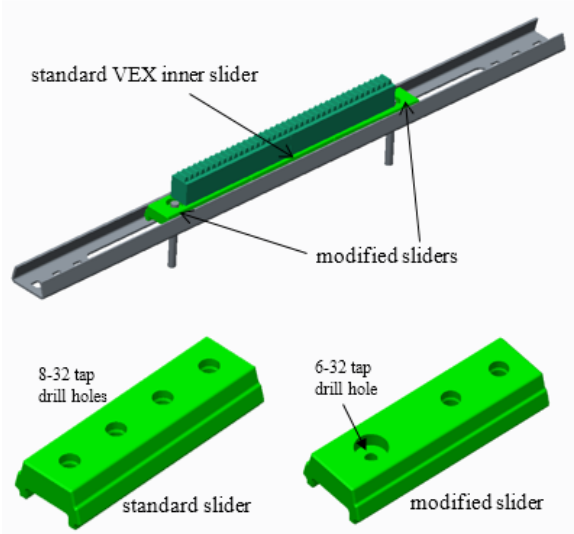


Figure 49. Control bar with rack on pan track (top); standard VEX inner slider (bottom left); modified slider (bottom right)

The control bar consisted of a standard VEX slider in between two custom sliders. The custom slider eliminated two of the holes of the standard slider, and replaced them with one counter-bored 6-32 tap drill hole. This hole was positioned on the custom slider such that the pin joints in the 6-32 holes would be 13.1 cm apart (the desired control bar length) when the sliders were placed in the specified series.

The hole size of the custom slider was chosen based on the design for the pin-joint linking the control bar to the control slider piece that slides on a track underneath the mechanism platform (Figure 50). Each of the two pin joints consisted of a 6-32 screw, which was self-tapped into the modified slider pieces of the control bar. The screw passed through the 8-32 tap drill clearance hole in the standard Vex slider piece below the platform. The head of the screw rested underneath the control slider, supporting the control slider and preventing the control track from falling (axial translation).

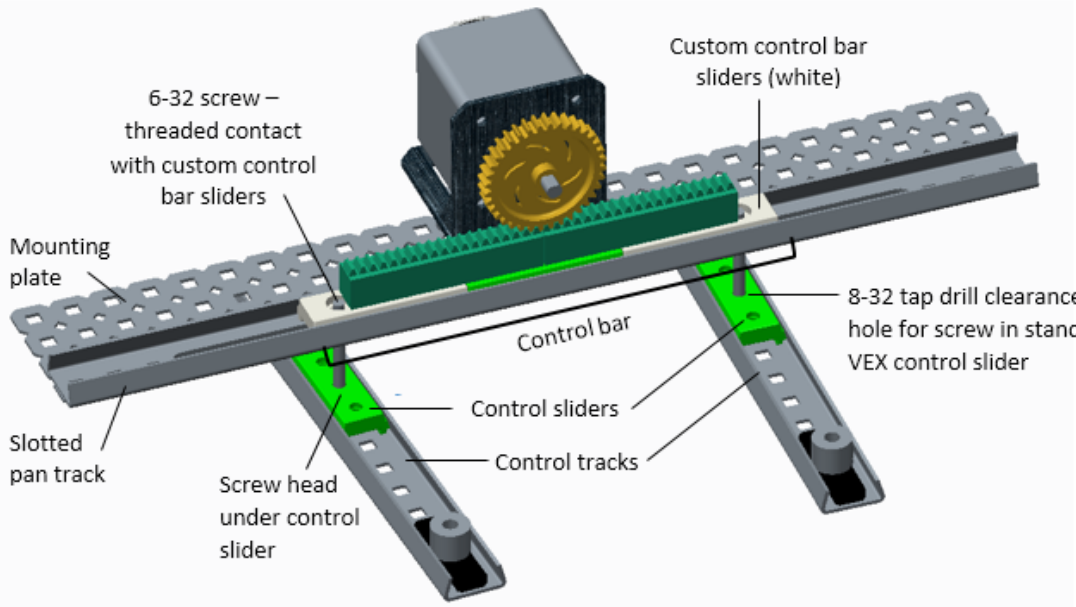


Figure 50. Pan Mechanism Assembly with Control Arms

9.2.3 Modified VEX Slider Tracks

The VEX slider tracks used for the control arms, the control bar, and control bar tracks all had to be modified in some way. The standard VEX linear motion sliders are produced in lengths of 12” and 17.5”. A single 17.5” slider track was cut with a hacksaw into two 8.5” tracks to create the control arms, shown as parts “A” in Figure 51. Similarly, another 17.5” VEX slider track was cut into two 6.5” tracks to produce the control bar side tracks, shown as part “B” in Figure 51.

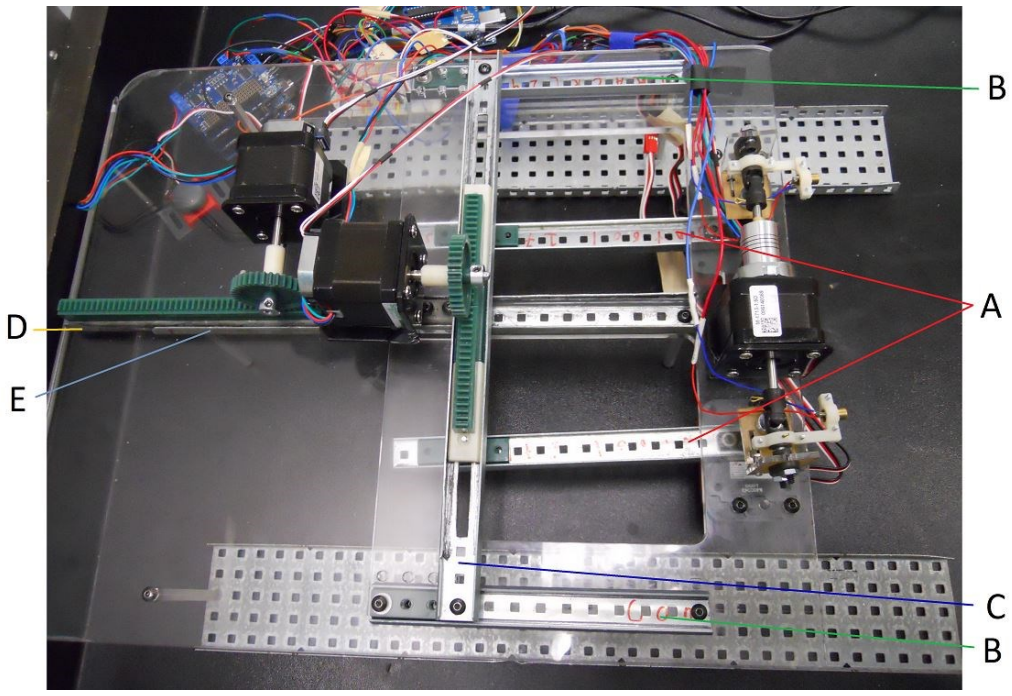


Figure 51: Modified VEX Linear Motion Slider Tracks

The pan mechanism slider track, part “C” in Figure 51, was cut down to 12.5” from a 17.5” VEX slider track. Additionally, two slots were cut into the slider track (see Figure 52) to allow the pin joints connecting the control bar, “C”, to the control arms, “A”, to pass through the bottom of the slider (see Figure 51). These slots were cut using a die grinder by cutting the material between nine of the standard holes already in the bar. These slots were 4.5” long each, and were separated from each other in the middle of the bar by three unaltered square holes.

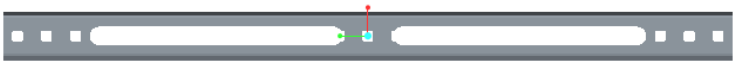


Figure 52. Slotted VEX slider track for pan mechanism

The convergence linear mechanism was composed of a VEX inner linear slider, “D”, and a VEX outer linear slider, “E”, each of a standard length of 12”. (Note that the convergence slider linkage in the assembled prototype consisted of the metal inner and outer tracks shown in Figure 51, as opposed to Delrin sliders in a metal track as previously modeled. This was changed to avoid interference between the exposed screw head mounting the outer slider track to the base, and the screw heads underneath the Delrin sliders mounting the racks to the sliders.) The outer slider was not modified, but the inner slider was cut to 10” so that it would not extrude beyond the edge of the platform base.

9.2.4 Laser Mount / Holder

The lasers indicating the lines of sight of the eyes had to be mounted to the U-joints for each eye. To accomplish this, custom laser eye mounts (or laser holders) were designed and manufactured. Each laser mount consisted of three parts that were printed using the Dimension Rapid Prototype

Machine with ABS plastic. The modeled parts and the full sub-assembly can be seen in Figure 53 and Figure 54 respectively. Part “C” depicted in Figure 54 connects the U-joint to the laser.

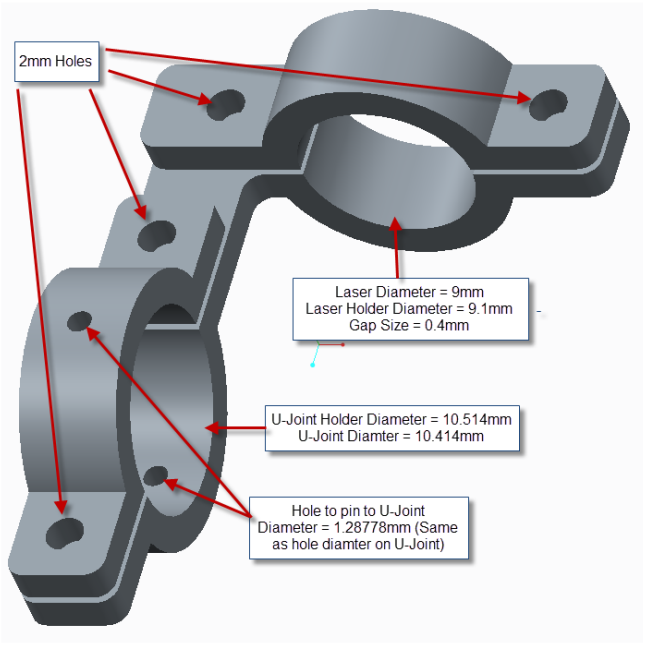


Figure 53: CAD Model of Laser Mount

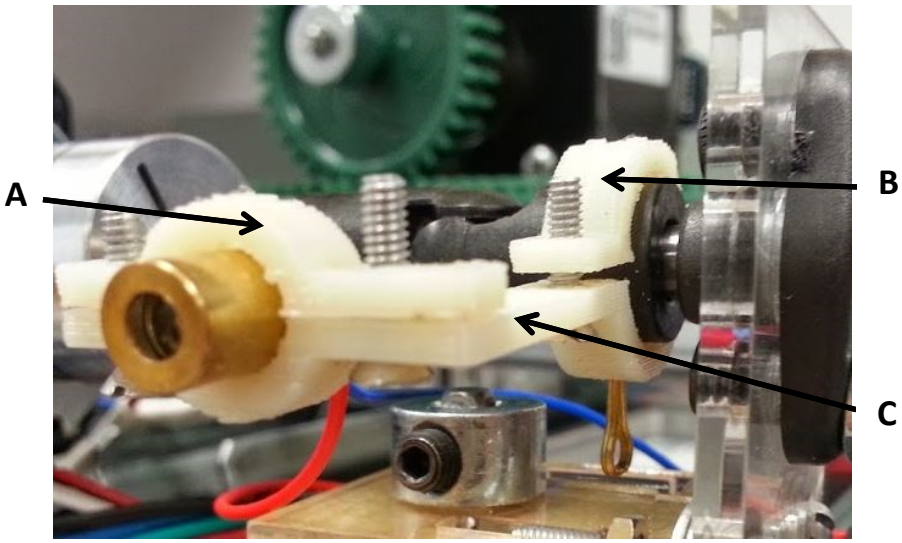


Figure 54: Assembled Laser Mount

The remaining two parts that made up the laser mount, “A” and “B”, were responsible for fixing the laser to the laser mount and fixing the laser mount to the U-Joint respectively. This was accomplished by the specific dimensioning of these two parts. First, the radii of the semi-circle, “A”, and the semi-circle, “B”, of the laser mount were approximately 0.001” larger than the radius of the U-joint and laser respectively. This was to accommodate for tolerances, ensuring that the part would fit around the U-joint and the laser. Second, each semi-circle extended approximately 175°. Therefore, there was a gap between two halves of the cylinder gripping the laser and the cylinder gripping the U-joint. Holes suitable for self-tapping screws pass through each of the three laser mount parts. #2 screws were used to fix the laser and U-joint to the laser mount. A 0.05” through-hole on the U-joint holder side of the

laser mount allowed a cotter pin to hold the laser eye mount, the U-joint, and the side tilt shafts in alignment.

9.2.5 L-bracket

A custom L-bracket, composed of two sheets of 0.25” acrylic and held together by two screws and nuts, was used to transfer the motion of the control arm slider tracks to motion of the U-joints making up the eyes. Figure 55 shows the L-bracket is rigidly attached to the square shaft that extended downwards to the control arm slider track. The square shaft is kept from being pulled down through the L-bracket by the shaft collar. Figure 55 also shows the tilt bearing on the vertical face, which was connected directly to the U-joint of the eye by a short shaft.

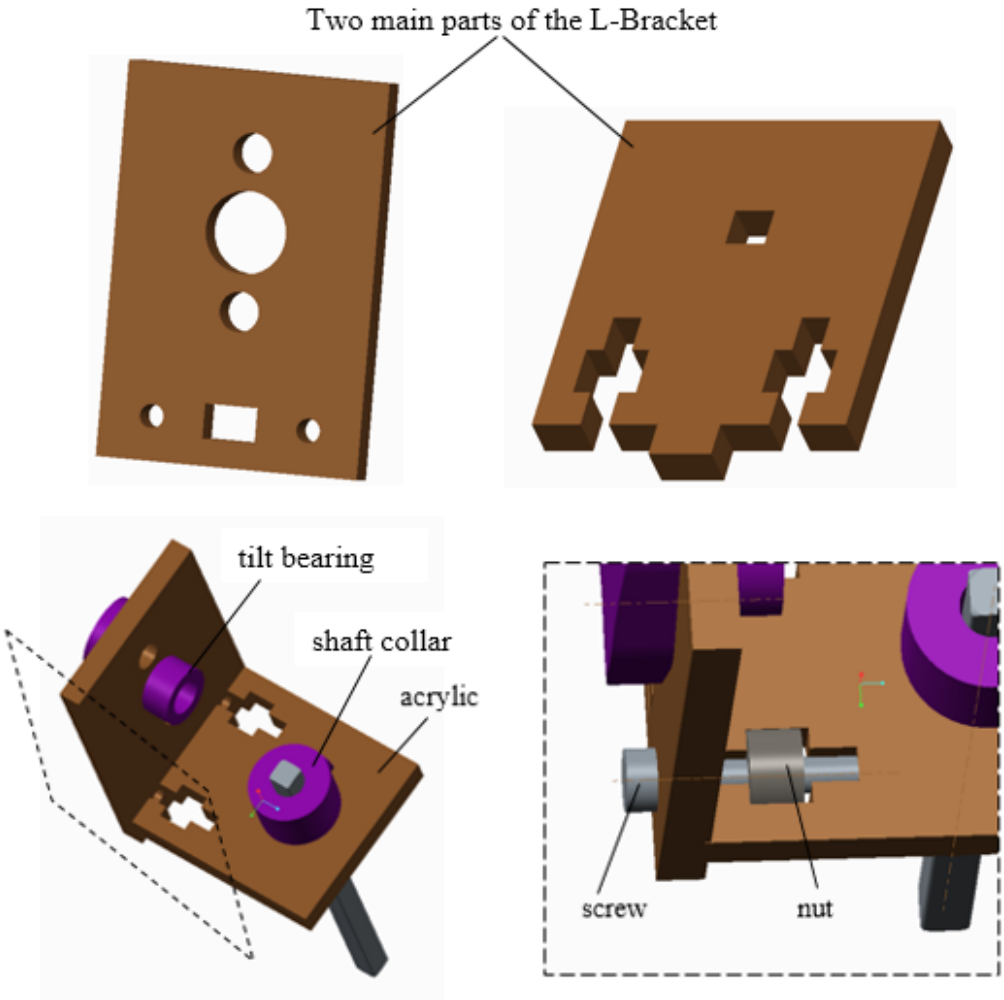


Figure 55: CAD Model of L-Bracket

9.2.6 Shaft adapter

Shafts of different sizes and shapes were used in the mechanism. Therefore, custom shaft adapters were manufactured to rigidly connect them to each other. The motor shafts had a D-profile, and were designed to connect to a standard VEX square shaft (0.125” side). Cylindrical parts - with a square hole on one end and a D-profile hole on the other - were 3D printed using the same rapid

prototyping machine that made the laser mounts (Figure 56). The holes in the part were designed so each shaft was inserted about 0.75 cm into either end of the ABS plastic part, utilizing an interference fit. An interference fit was used to eliminate play and backlash when the motor was to be actuated.

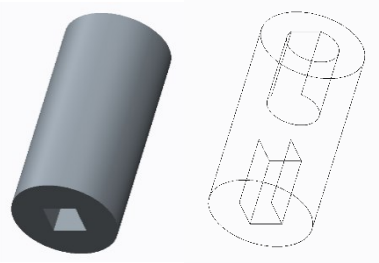


Figure 56. CAD model of 3D printed VEX shaft-to-motor shaft adapter

9.3 Assembly

Full assembly of the mechanism began once all the parts were acquired and manufactured. Much of the assembly process is self-explanatory, especially given the description of each purchased and manufactured part in Section 9.1 and Section 9.2. However, a brief summary of the assembly process is provided below, along with a discussion of the main issues encountered.

Standoffs, square shafts, shaft collars, lock plates, spacers, screws and nuts were acquired from lab stock on campus.

First, the main convergence track and the two side slider tracks were screwed onto the acrylic base. Next, standoffs were screwed onto the base, elevating it off the ground. The mounting plate for the pan mechanism and the slotted slider track for the control bar were screwed onto the two sliders that go in the side-support vergence tracks mentioned previously. The two control bar VEX racks were screwed onto the three sliders (including the two modified sliders on each end), completing the top portion of the control bar, which was inserted into the pan slider track. Next, the two screws (the pins for the pin joint) at the ends of the control bar were inserted into the modified sliders. Three spacers were added to each screw, then the screws were screwed into the control bar from underneath the platform. The pan stepper motor was mounted to the pan mounting plate and the shaft adapter was used to connect the motor shaft to the square shaft holding the pan gear. The position of the pan stepper motor was adjusted so that the gear was correctly aligned with the control bar rack. The convergence motor was mounted to the platform and the convergence gear was attached to the motor in a similar fashion so that it aligned with the convergence rack. Absolute encoders were mounted on the back side of each stepper motor and the control system was set up, as discussed further in Section 9.4. This completed the main assembly of the pan and convergence portion of the prototype.

When inserting the sliders onto the lower control tracks below the base, it was necessary to insert the sliders into the tracks before mounting the control arm tracks to the square shaft below the center of each eye. Once these control arm tracks were in place, all of the parts along the square shaft below the eyes were assembled. These parts are shown in Figure 57. Specifically, the square shaft was inserted through the platform, shaft collar, lock plate, control arm track, and encoder. Each lock plate was fixed to the control arm track using two screws and the shaft collars located directly above each mounting plate were tightened. The two parts of the acrylic L-bracket were screwed together and each L-bracket was mounted on the square shaft. Another shaft collar was used on each eye on the square shaft directly above each L-bracket.

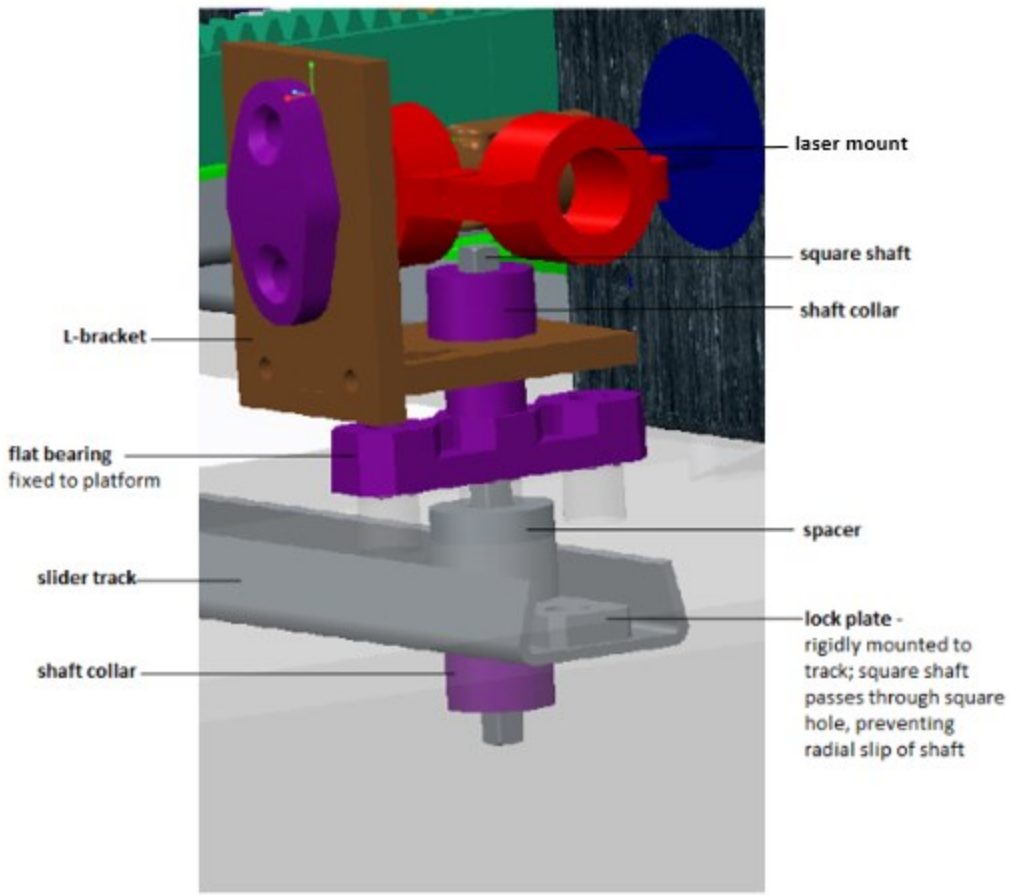


Figure 57: Assembly of "Eye", represented by the laser mount, and parts along the square shaft below it

Next, the eyes and tilt mechanism were assembled. The tilt motor with attached encoder was mounted to the platform between the two eyes. The appropriate tilt shafts, which were previously cut to the correct lengths, were inserted into each side of the tilt motor. The tilt shaft in the flex coupler on the side of the tilt motor near the left eye was tightened around the shaft so that it extended the appropriate amount such that the center of the left U-joint would be directly above the square shaft. A small amount of tape was used to encircle the tilt shaft before it was inserted into the flex coupler so that it had a large enough diameter to be properly gripped by the flex coupler. Cotter pins were used to fix the U-joints to the left and right tilt shafts. The end shafts were inserted into the other side of each U-joint. The other side of each of these end shafts was then pin-jointed to the tilt bearings (shown in purple in Figure 57) on the vertical face of the L-brackets. The lasers were mounted to the laser holders by screwing them into place, then the laser mounts were pinned to the U-joints and finally screwed into place. A photo of the tilt mechanism and eye assembly is shown in Figure 58.

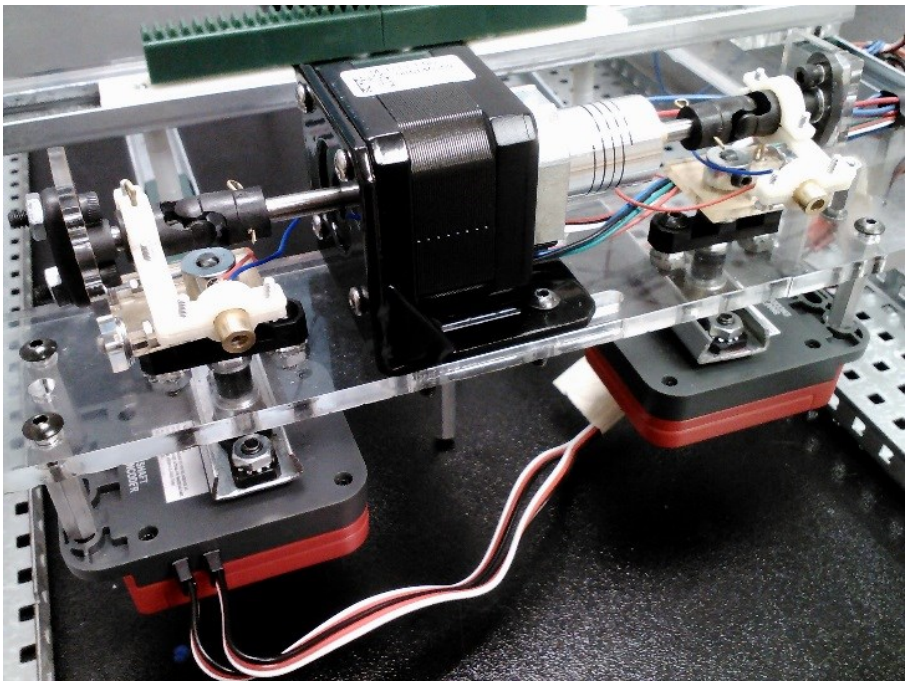


Figure 58. Assembled tilt mechanism, L-brackets, universal joints and laser-pointer eye components

After the platform was assembled, multiple adjustments were made to reduce friction and backlash. The convergence slider track had to be slightly widened manually using pliers to increase the space for the slider and decrease the friction. The sliders and tracks were also lubricated to reduce friction. There was significant backlash between the square shafts below the eyes and the L-brackets. To solve this problem, L-brackets with smaller holes were manufactured to make the fit between the square shaft and square hole in the L-bracket tighter. To reduce backlash between the square shaft and the control arm track, super glue was placed around the square shaft at that location. Lastly, to reduce misalignment between the laser holder and the control arm track, the lock plate connecting the square shaft to the control arm was adjusted by rotating it slightly as appropriate and then tightening it again. The fully assembled prototype is shown in Figure 59.

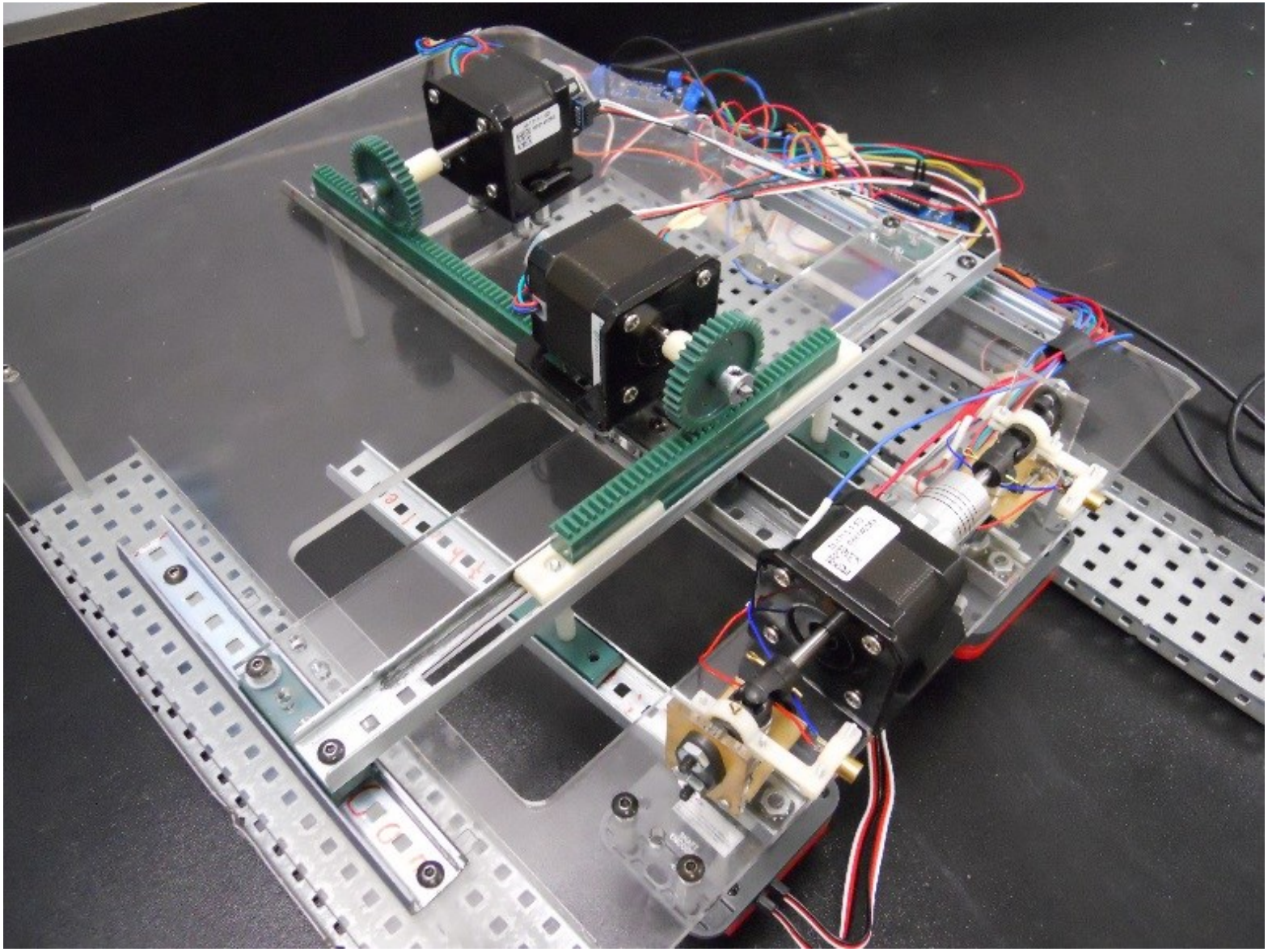


Figure 59: The Fully-Assembled Prototype

9.4 Implemented Control System Architecture

The control system architecture served two main purposes – control of the stepper motors to get to the desired focus point positions and measurement of the eye pan velocity and acceleration.

The parameter to be controlled was the focus point position in the x-y-z coordinate system. While it was desirable to control the velocity and acceleration of the focus point, the simultaneous control of the focus point position, velocity and acceleration is a more advanced control process and was beyond the scope of this project.

There were two possible control options to get the focal point to the desired position. The first method was to use a closed loop feedback control by detecting the focal positions and determining the motor input based on the feedback signal of focal positions. To use this method, it was important to determine at which part of the mechanism the feedback signal would be measured. The eye orientation was the parameter that gave the focal position with the least amount of error, assuming that the eye orientation is measured with respect to ground. Nevertheless, measurement of focus point position through the orientation of the eyes retained error from the laser mount and universal joint. The ideal measurement would be the focal position itself but it was impractical to mount a sensor on the surface on to which the laser pointer is projected. Such a sensor could also not be used in a practical application.

The second control method was open loop feedforward control. This was simpler because it involved running the stepper motor without feedback from the mechanism. Each motor could be programmed to rotate a certain number of steps (calculated using the kinematic equations developed in (Appendix A: Generating System of Equations to Determine Input and Output Design Parameters), resulting in the desired pan, convergence and tilt movements. Similar to the previous method, this method would have ignored any error resulting from the mechanical transformation of the stepper motor position to the actual focus point position output.

Despite such drawbacks in mechanical error from the latter method, open loop control was chosen as the final control design. In order to make closed loop control work effectively, the measurement of feedback needed to be as accurate as possible. This would require redesigning our laser mount and universal joint. Since design reiteration could not be performed due to time constraints, implementing open loop control would be just as effective as closed loop control. As discussed above, each stepper motor has a predefined number of steps per revolution which makes them rotate and stop exactly at the desired steps. However, in some applications which require high torque or speed, stepper motors may skip some steps while rotating. Due to minimal payload properties of our mechanism design, motor stall is not expected to happen during operation.

Figure 60 shows the complete control system. An Arduino Uno microcontroller was the central control unit of the system. Two Adafruit Motor Shields were attached to the Arduino Uno for motor control. The Arduino Uno was powered by 5 V from the computer to which it was tethered. The motor shields received 12 V from an external power supply to power the motors. The inputs provided to the Arduino were steps (position), maximum speed (the speed the motor can handle without losing any steps) and constant acceleration. The motor would ramp up to half of the steps with the specified acceleration and then decelerate until it completed the remaining steps. From the kinematic equations explained in Appendix A: Generating System of Equations to Determine Input and Output Design Parameters, pan, tilt and convergence movements can be related to the focus point position expressed in x-y-z coordinates. These linear movements were related to inputs of motor steps. This was, therefore, an open loop motor control of a forward kinematics problem, since stepper motors were found to accurately follow feedforward drive commands. The encoders mounted at the stepper motors and the eyes gave voltage signals which were used to calculate rotary position of each of the corresponding

shafts. The internal clock in the Arduino was utilized to differentiate position changes against time to calculate velocity and acceleration signals.

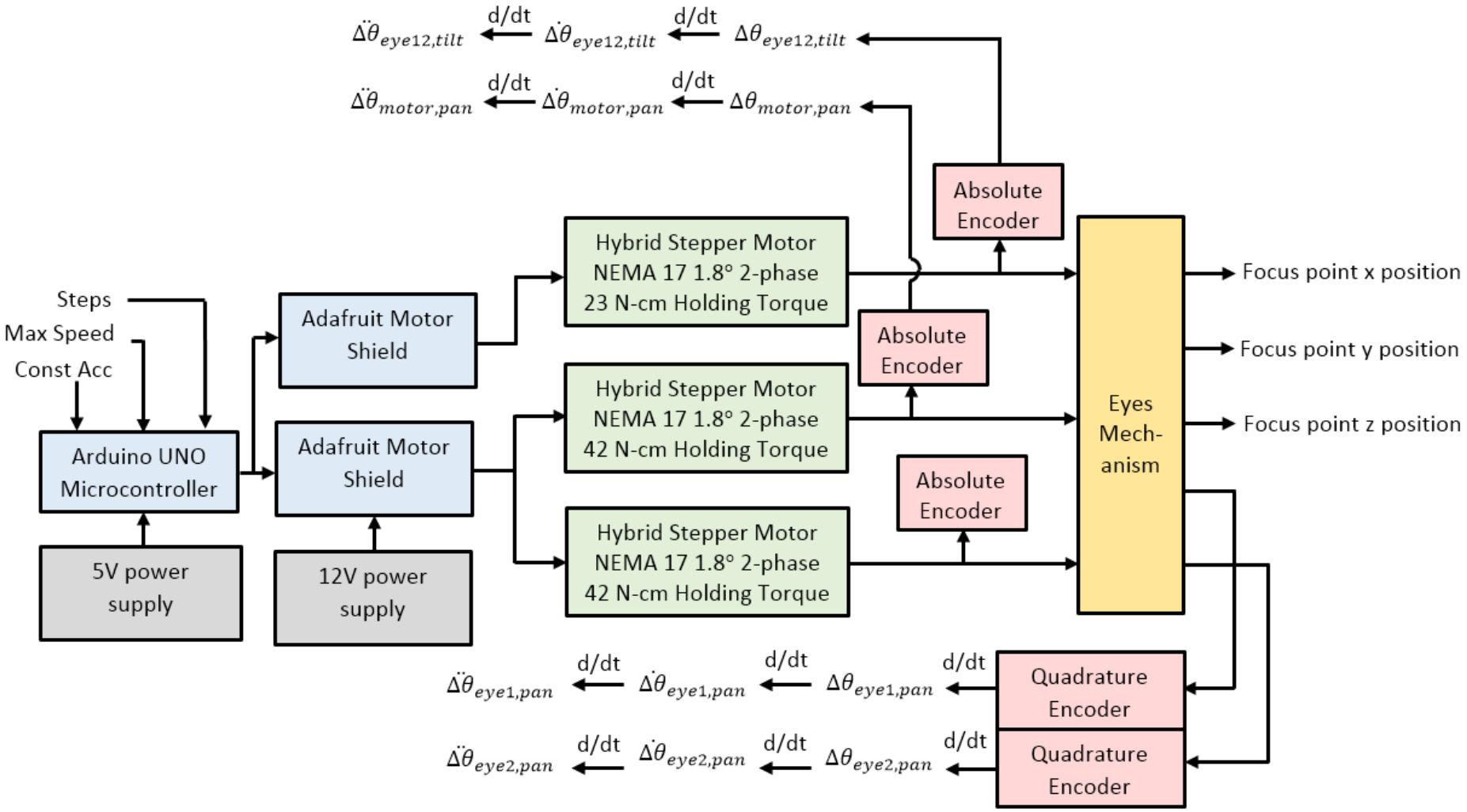


Figure 60: Control System Architecture of Platform Prototype

10 Testing

Various tests were conducted to evaluate how well the prototype mechanism met the design specifications. The following sections describe the procedures followed and results obtained from the focus point position accuracy tests and the maximum eye velocity and acceleration tests.

10.1 Focus Point Position

The objective of the position testing is to calibrate the mechanism and determine whether the lines of sight of the two eyes are on the same plane and will converge, as well as to determine the position error between the desired input position and the actual position of the focus point.

10.1.1 Experimental Setup and Procedure

The mechanism platform was clamped in place on a stable, level podium. The podium was positioned in front of a flat vertical whiteboard such that the eyes were equidistant from it and the lasers would point onto the board (see Figure 61). The appropriate distance from the eyes to the board was chosen so that at 45° and -45° tilt angles the laser points would intersect the board and could be measured (the eyes were placed approximately the same distance from the board as the distance between the baseline and the bottom of the board). The exact distance between the eyes and board was measured to be 52.5 cm, and was recorded as variable (*a*) (see Figure 62).

The mechanism was set to an initial position (with a theoretical focus point of zero tilt, zero pan and the minimum convergence depth of 75 cm) and the distance between the laser points on the board was checked to ensure that the lines of sight were converging. Because (*a*) was less than the convergence depth in this case, the focus point was expected to be located behind the face of the board and therefore the distance between the two points should be less than the baseline between the two eyes (*b* = 12.5 cm). Additionally, if the lasers are converging behind the board, then the laser on the left eye should be the leftmost point of the two points intersecting the board (if the focus point is in front of the board the lasers cross before intersecting the board and the left eye laser will hit the board to the right of the right eye laser). This can be checked by covering the right laser to see which point on the board is coming from the left eye. These methods were used repeatedly throughout testing to check against misalignment.

The mechanism was calibrated (by making small adjustments to the control bar, lasers and laser holders) such that the laser points were on the same horizontal plane. This horizontal plane was marked on the board as the x-axis, and the y-axis was also marked based on the mid-point of the laser points when at zero pan position (see Figure 61).

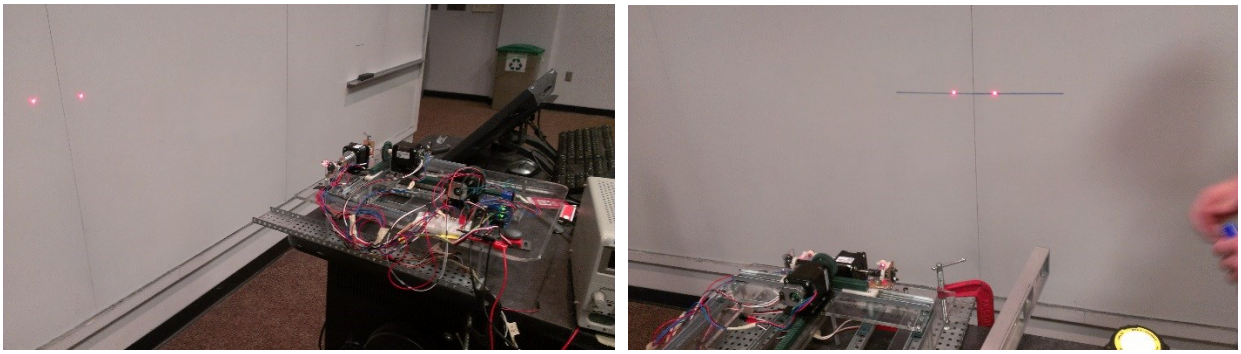


Figure 61: Experimental setup for position error testing (photos above are from an initial test setup when the distance from the board was greater than 52.5 cm)

The mechanism was moved to several different positions for measurement. First, the tilt angle was set to zero and the convergence depth was held at the constant minimum value of 75 cm. The eyes were then panned, starting at zero pan and then panning to the left and right limits—in order, the pan positions (*fx*) tested were 0, -104 and 104 cm (relative to the y-z plane). From the far-right pan position (theoretically at 104-cm-pan), the vergence position was then changed to the predicted midpoint convergence depth of 178 cm. The panning procedure was then repeated—the mechanism was returned to zero pan, and then panned to the extreme left and right. This process was again repeated for the maximum convergence depth of 273 cm. After completing the combinations of the midpoint and limit positions of pan and convergence at zero tilt, the tilt motor was used to tilt the eyes upwards 45° and the convergence and pan procedure described was repeated.

To calculate the actual location of the focus point, three manual measurements were made for each test position: the horizontal distance from the left-eye laser point to the y-axis (*Lx*, positive to the right of the y-axis and negative to the left), the horizontal distance from the right-eye laser point to the y-axis (*Rx*, following the same sign convention as *Lx*), and the vertical distance from the laser points to the x-axis (*h*, positive above the x-axis and negative below), as shown in Figure 63. (Note that only one tilt value *h* was recorded because both two laser points consistently had the same height.) The following equations were used to calculate the actual focus point positions from the measurements gathered during testing:

$$fz_{actual} = \frac{b \times a}{(b - Rx + Lx)}$$

$$fx_{actual} = \frac{fz \times (Lx + Rx)}{2 * a}$$

$$\theta_{tilt,actual} = \tan^{-1}\left(\frac{h}{a}\right) \times \frac{180}{\pi}$$

where constant inputs *b* = 12.5 cm is the baseline between the two eyes, and *a* = 52.5 cm is the distance from the eyes to the whiteboard.

These equations were based on trigonometry and similar triangles; see Appendix G: Derivation of Equations for Calculating Focus Point Position for Position Error Testing to see their derivation.

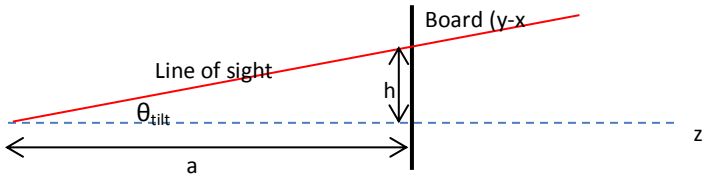


Figure 62. Diagram illustrating parameters for calculating tilt angle

The parameter *fz* is proximal depth of the focus point (in z-direction) projected onto the x-z plane, measured from the baseline of the eyes. The parameter *fx* is the lateral distance of the focus point projected onto the x-z plane, measured from the z-axis (see Figure 63).

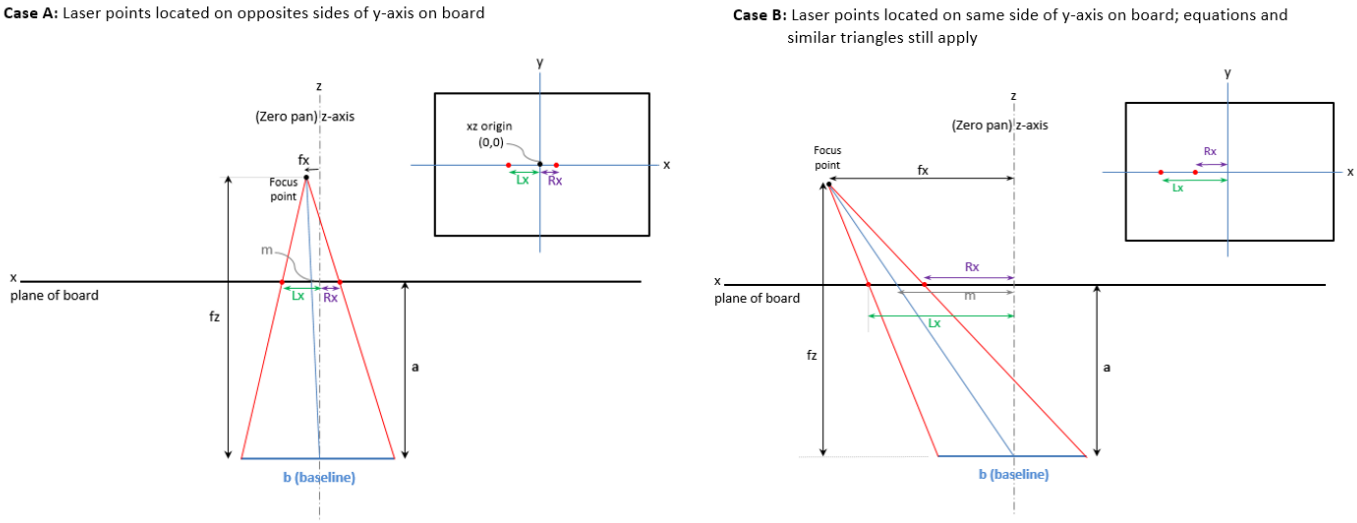


Figure 63: Diagram illustrating parameters for calculating actual pan and convergence positions of focus point

10.1.2 Arduino Programming for Focus Point Position and Range of Motion Testing

Although the absolute encoders mounted on the stepper motors were initially intended to be used for calibration, the process of using Serial Peripheral Interface (SPI) to determine absolute encoder reading required knowledge of embedded computer engineering. This was beyond the scope of the project and this calibration was left out in the final control system design. Therefore, only incremental encoder values were obtained from the absolute encoders in the mechanism. These values were used to attain speed and acceleration of the pan and tilt motors.

The mechanism was manually calibrated to the original configuration in which the control bar was placed in the center of the pan slider track at the minimum convergence depth. The desired motor steps for pan and convergence were calculated using the equation below:

Step Equation

$$\theta_{step} = \frac{s}{r_{pinion}} \times \frac{200}{2\pi}$$

s = pan track movement (cx)/ convergence track movement (cz)

r_{pinion} = pitch radius of the Vex pinion = 1.905 cm

To be able to send commands of steps at a desired speed and acceleration by using functions, the existing library called <AccelStepper.h> was used for convenience. Two libraries from Adafruit called

<Adafruit_MotorShield.h> and "utility/Adafruit_PWMServoDriver.h" were also used to be able to run and control three stepper motors simultaneously from the stepper motor driver.

For position testing, maximum, minimum and median changes in value of pan and convergence track movements from the original configuration were entered into variable s of the Step Equation to find the desired steps in pan and convergence motor respectively. On the other hand, tilt angles of 45 deg and -45 deg were related to motor inputs of steps by motor resolution of 200 steps/360 deg.

Steps in the motors that correspond to desired pan, tilt and convergence movements were declared as arrays X, Y and Z of three step values respectively in Uno as shown in Figure 64.

```
int X[3]={
  0,86,-86};
int Y[3]={
  0,-25,25};
int Z[3]={
  0,95,190};
```

Figure 64: Step Arrays

The index of each array was put into the loop to run the motor to 27 set focus positions caused by combinations of three variations in each pan, convergence and tilt movement. Figure 65 shows the step input to the tilt motor in a loop. TiltStepper.moveTo(Y[i]) moved the tilt stepper motor at the defined velocity and acceleration from “setup()” by one of the steps declared in array Y. TiltStepper.runToPosition () moved the tilt stepper motor by desired steps until all the step rotations were complete. The loop was also used for the pan and convergence motors. During the transition from one loop to another i.e, when the mechanism moved from one test point to another, the mechanism waited for the user to press a button switch attached to Uno. This allowed the focus point position test data to be manually measured on the board on which the lasers were being projected (see Section 10.1.1)

```
for(int i=0;i<3;i++){

  TiltStepper.moveTo(Y[i]);
  TiltStepper.runToPosition();
  Serial.println(TiltStepper.currentPosition());
  // wait for button press before proceeding
  while(i != 0 && digitalRead(buttonPin) == HIGH) {}
```

Figure 65: Example Code for the Loop of Tilt Motor Steps

The complete code used for focus point position testing is included in Appendix E: Arduino Code for Focus Point Position Testing.

10.1.3 Results

Originally, it was planned to test twenty-seven possible combinations of three selected values for each degree of freedom. For pan and tilt, the selected values were the theoretical limits and midpoints of the mechanism’s range (0, 104 and -104 cm for pan and 0, 45° and -45° for tilt); for convergence, the selected depths were the theoretical minimum, midrange and maximum values (75, 178 and 273 cm, respectively). Upon beginning testing, it was found that the mechanism is very sensitive to misalignment in the laser holder or control bar, and the initial tests had an unexpectedly large amount of position error. With a better idea of the specific sources of error and how to better adjust the mechanism and prepare for testing using the procedure described in Section 10.1.1, the position test was repeated for 15 points. For these tests, symmetry across the x-z plane was assumed for tilt and only 0° and 45° tilt angles were tested; for 0° tilt, the limits of range and midpoints were tested for pan and convergence, and for 45° tilt only the limits of pan and convergence were tested. Due to time

constraints, one successful trial was completed for each test point. There were no interruptions or adjustments made mid-test.

The tested theoretical points, corresponding actual focus point positions, and the linear error between them are listed in Table 7. The "Pan Angular Error" of the points is the difference between the pan angles of the actual and theoretical focus points; the "pan angle" of a point is defined as the angle of the line from the midpoint of the eyes to the point projected onto the x-z plane, measured from the z-axis. This line is the blue line between the lines of sight shown in the diagrams of Figure 63.

Table 7: Position Error Test Results

Input (Theoretical) Positions			Actual Focus Point Position						
Pan, fx (cm)	Convergence, fz (cm)	Tilt (deg)	Actual fx (cm)	Error in fx (cm)	Pan Angular Error (deg)	Actual fz (cm)	Error in fz (cm)	Actual Tilt (deg)	Error in tilt (deg)
0	75	0	0	0	0.0	72.1	-2.9	0	0
-104	75	0	not measureable due to motor interference						
104	75	0	97.6	-6.4	-7.0	90.5	15.5	0.0	0.0
0	178	0	1.8	1.8	0.5	187.5	9.5	-1.3	-1.3
-104	178	0	-303.6	-199.6	1.3	546.9	368.9	0.0	0.0
104	178	0	91.4	-12.6	-1.2	164.1	-13.9	0.0	0.0
0	273	0	1.6	1.6	0.3	285.3	12.3	0.0	0.0
-104	273	0	-80.0	24.0	3.9	262.5	-10.5	0.0	0.0
104	273	0	62.6	-41.4	-2.1	184.9	-88.1	0.0	0.0
0	75	45	2.0	2.0	1.9	59.7	-15.3	42.4	-2.6
-104	75	45	-100.0	4.0	1.9	77.2	2.2	42.4	-2.6
104	75	45	72.6	-31.4	-4.9	62.5	-12.5	46.3	1.3
0	273	45	-1.8	-1.8	-0.5	187.5	-85.5	41.2	-3.8
-104	273	45	-69.8	34.2	3.2	218.8	-54.3	42.7	-2.3
104	273	45	53.9	-50.1	-2.7	164.1	-108.9	40.6	-4.4

Plots showing the convergence and pan positions (effectively the x-z plane, a top view of the field of vision) for the two tilt angles tested are shown in Figure 66 and Figure 67. The points are numbered to indicate the corresponding theoretical and measured positions, as well as the order in which the points were tested. (Note that the points for position 2 in Figure 66 are missing because the laser points could not be measured due to motor interference.)

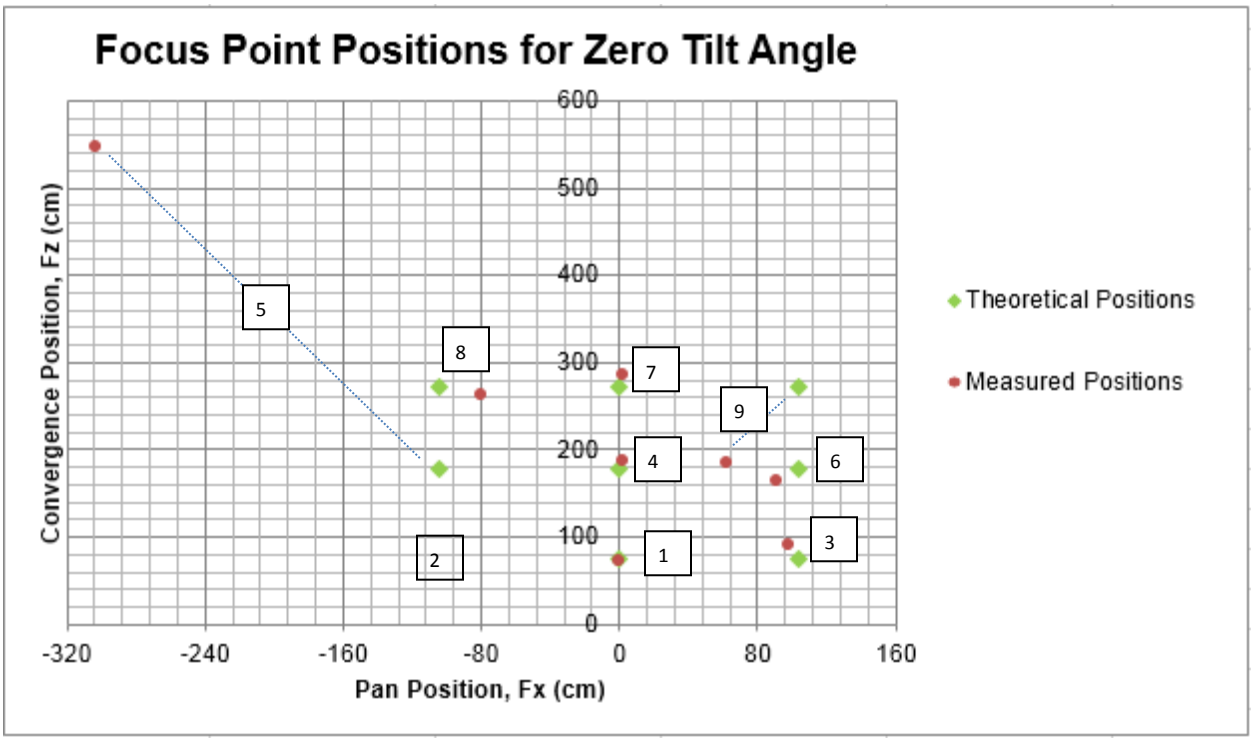


Figure 66: Plot of Pan and Convergence Position Error Results for Zero Degrees of Tilt

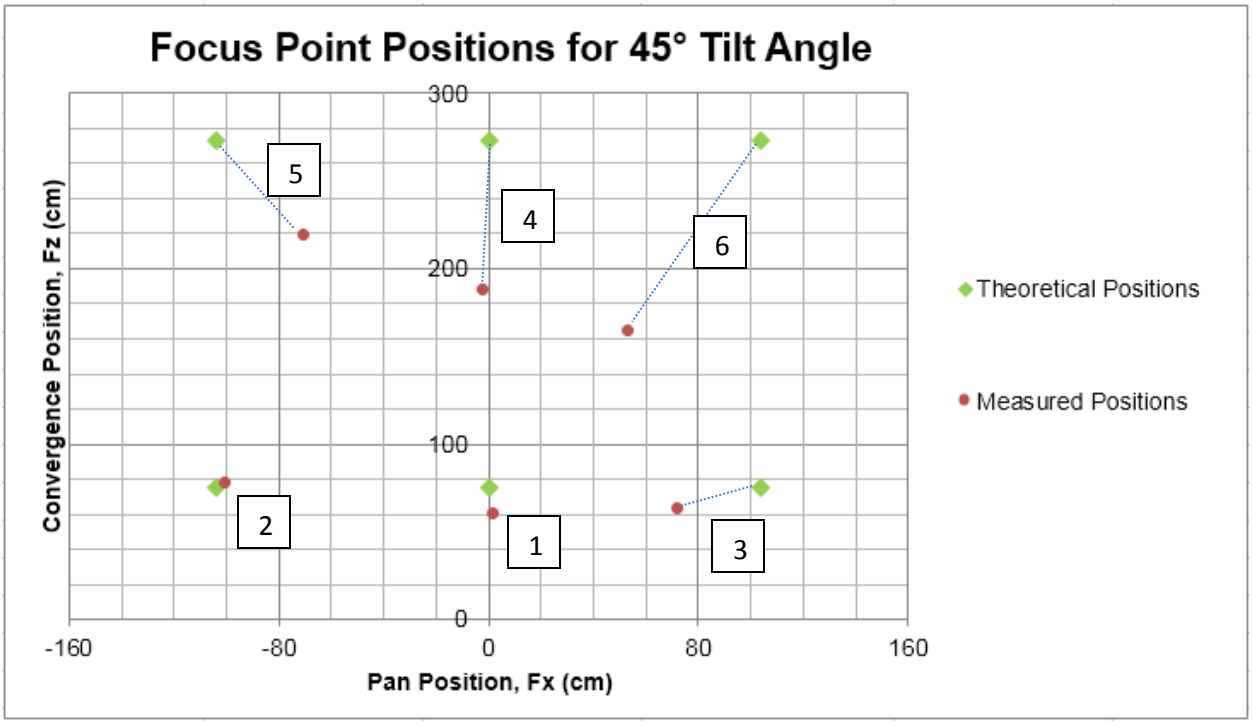


Figure 67: Plot of Pan and Convergence Position Error Results for 45 Degrees of Tilt

10.1.4Analysis

As seen in the pan vs. convergence plots for 0° and 45° tilt angles, the smallest linear error tended to occur at zero pan positions and was minimized when the convergence depth was small. For example, excluding the major outlier, the average deviation of the measured value from the theoretical was 25.11 cm for *fz* and 1.43 cm for *fx* when pan was 0. The average of the differences for all other non-zero pan positions was 42.33 cm for *fz* and 34.23 cm for *fx*. For the data taken at minimum convergence depth, the average deviation from the theoretical was 9.69 cm for *fz* and 8.75 cm for *fx*, while the average deviation for data taken at the maximum convergence was 59.94 cm for *fz* and 25.52 cm for *fx*. When the tilt angle was changed from 0 degrees to 45 degrees, the average linear error in *fz* increased noticeably, but the average linear error in *fx* did not change significantly. The linear error ranged greatly, from 1.8 cm of error to as high as 368.9 cm when the mechanism was unexpectedly misaligned. However the pan angle error remained relatively small. The pan angle error also does not seem to be constant or to have a constant trend.

Overall, it appears that the linear and angular position error is not constant. During testing, it was observed that much of the inconsistency was due to mechanical misalignment. The results were very sensitive to even very small misalignments in the mechanical components, which yielded some outlier data with high error. For example, the laser holder was subject to some misalignment relative to the slider tracks below the platform, which are designed to be parallel to the lasers' lines of sight. This may be due to the fact that the laser holder was a separate component from the universal joint, giving rise to the potential for mounting inconsistencies. It is also possible that the stepper motors are not perfectly consistent, and gear backlash may also be contributing to error in the amount of control bar translation. Gear backlash may be partially responsible for errors at the limits of motion, which were tested when the control bar actuation was changing direction. These changes in direction may also have caused the effective control bar length (the distance between the pin joints on the control sliders beneath the platform) to change slightly, due to pin misalignment and manufacturing tolerances in the sliders. At this time, however, there is not enough data to conclude that there is a trend in the amount of error based on the order in which the mechanism moves to each point and the resulting direction changes of the control bar. Human error may also have been a factor in making the manual measurements used for calculating the actual focus point positions.

Our original design specifications for linear resolution were 1.2 cm for pan, 1.2 cm for vergence, and 1.8° in tilt angle. For pan and vergence, most of the linear error was greater than our desired resolution. As seen in Table 7, the smallest linear error achieved for the pan position was 1.63 cm, 2.2 cm for vergence, and 1.3° for non-zero tilt angle. However, the pan angle error was mostly very small - 2.25° on average - even for the data points with high linear convergence error (observed especially for 45° tilt). A small pan angle error is important for making a robotic vision system as realistic as possible when interacting with a human.

A calibration analysis was conducted based on the experimental data. A system of equations was derived in which the inputs for each tested data point were the experimental pan position fx_{actual}, experimental pan angle, constant baseline length *b,* experimental convergence depth fz_{actual}, and corresponding control bar positions *cx* and *cz*. The effective actual control bar length l_c for each point, as well as the effective angular misalignment $\theta_{misalign}$ (parameters that, in previous numerical analyses, had been assumed to be constant inputs or be equal to zero), were unknowns that were calculated for each data point. See *Appendix H: Position Calibration Analysis* for the system of equations. These parameters were chosen to be treated as the unknowns because they were most likely to be causing the large position error in the test results; doing a backwards calculation to determine how these parameters

differ between the prototype and the numerical analysis of an ideal mechanism would allow better calibration of the prototype.

The results of the calibration analysis for each data point are shown in Table 8. The calibration parameters (control bar length and angular misalignment) were found to be variable. This variation could be due to additional sources of error such as backlash that were not yet incorporated into the calibration analysis. The average backwards-calculated control bar length was 13.2 cm (compared to the ideal length of 13.1 cm) and the average angular misalignment was -0.3° (compared to the ideal 0 offset). For future work with the constructed prototype, the calibration parameters can be incorporated into the model and controls to reduce position error.

Table 8: Calculations of calibration parameters (control bar length and effective angular misalignment) for each data point

Input (Theoretical) Positions				Calibration Parameters	
Pan, fx (cm)	Convergence, fz (cm)	Tilt (deg)		Calculated control bar length (cm)	Angular Misalignment (deg)
0	75	0		13.12	0.00000
-104	75	0		not measureable	
104	75	0		13.00	-0.12400
0	178	0		13.06	0.00953
-104	178	0		12.69	-1.04400
104	178	0		13.1	-0.02900
0	273	0		13.1	0.00571
-104	273	0		13.1	-0.66000
104	273	0		13.4	-0.03800
0	75	45		13.3	0.03300
-104	75	45		13.1	-1.86000
104	75	45		13.2	-0.08700
0	273	45		13.4	-0.00953
-104	273	45		13.2	-0.30900
104	273	45		13.5	-0.04700
			AVERAGE	**13.2**	**-0.3**

10.2 Eye Velocity and Acceleration

The objective of eye velocity and acceleration testing is to evaluate the mechanism eye velocity and acceleration against a minimum human saccadic eye speed (500 deg/s) and acceleration (25,000 deg/s^2).

10.2.1Procedure

The mechanism was run using the Arduino program described in Section 10.2.2 (see Appendix F: Arduino Code for Eye Pan Velocity, Acceleration Testing). The mechanism was manually calibrated to the original configuration where the control bar was placed in the center of the pan mechanism slider track at the minimum convergence depth. The control bar was then panned from the original configuration to the extreme right and left and then back to original configuration such that the focus point moved from 0 to -104 cm, 104 cm and then back to 0 successively. This was tested for minimum, median and maximum convergence depths of the control bar which correspond to focal depth at 73 cm, 178 cm and 273 cm respectively from the baseline. This procedure can be understood better from Figure 68 which shows the order of focus point positions.

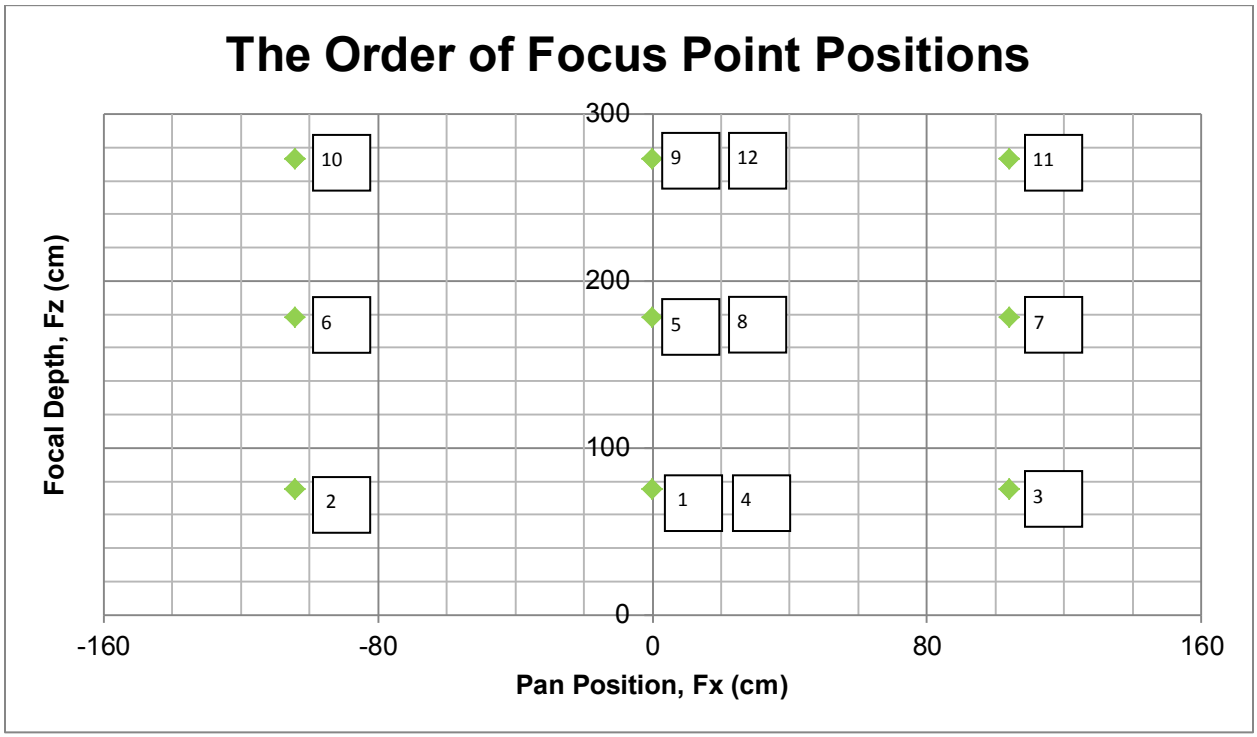


Figure 68: The Order of Focus Point Positions used in Eye Velocity and Acceleration Testing

The data were recorded in real time by another Arduino Uno (Arduino B) simultaneously in real time. In addition, the data was imported into Microsoft Excel so that it could be stored and analyzed. PuTTY, a free and open-source serial console application, was used to connect to the Arduino B serial port and import the data to Excel in real time.

The data imported into Excel are old and new encoder count values from the left and right eyes and the pan stepper motor, and old and new timer counts from the Arduino B. The Vex encoder has a resolution of 90 pulses per revolution. Since the resolution was increased by using an interrupt, 180 encoder counts made up one revolution. To convert the Vex encoder counts to actual square shaft

rotation in degrees (pan angle of each eye), the count was multiplied by two. On the other hand, the absolute encoder mounted at the pan stepper motor has a resolution of 1024 pulses per revolution. The use of an interrupt increased the resolution to 2048 pulses per revolution. Pulse counts from the pan stepper motor were multiplied by 360/2048 to find stepper motor rotation in degrees.

To calculate velocity, the change in angles was divided by the corresponding time interval in Excel. The eye velocity was represented as below:

$$v = \frac{(count_{new} - count_{old}) * 2}{time_{new} - time_{old}}$$

Similarly, to calculate acceleration, the change in velocity was divided by the corresponding time interval in Excel. The eye acceleration was represented as below:

$$a = \frac{(v_{new} - v_{old})}{time_{new} - time_{old}}$$

10.2.2 Arduino Programming for Eye Pan Velocity, Acceleration Testing

The mechanism needed to be actuated to test the eye velocity at different eye configurations (see Section 10.2.1). The pan motor was given inputs of steps at a certain constant speed to get the control bar to the extreme left and right and the convergence motor was also given a certain number of steps and constant speed to get the control bar to minimum, medium and maximum convergence depth. Figure 69 shows the example code for achieving the sequence of pan and convergence motor actuation by using a dummy variable called "i" which was initialized as zero. In the example code, when "i" was zero, the "while" loop rotated the pan motor such that the control bar was translated all the way to the right (focus point translated to left). After this actuation had been executed, the dummy variable "i" was changed to enter the while loop of "i" as "1" which drove the pan motor such that the control bar was translated all the way from the right to the left. This sequence of actuation was carried out to drive the convergence motor as needed for testing.

```
while (i==0){
      PanStepper.moveTo(86);     // write the steps
      PanStepper.setSpeed(500.0);
      PanStepper.runSpeed();
      if(PanStepper.currentPosition()==86)
      i=1;
    }

while (i==1){
      PanStepper.moveTo(-86);     // write the steps
      PanStepper.setSpeed(-500.0);
      PanStepper.runSpeed();
      if(PanStepper.currentPosition()==-86)
        i=2;
      }
```

Figure 69: Example Code to Rotate Pan Motor in Steps in the Desired Order

Due to the way the functions from the library worked, the motors were not able to rotate smoothly when the encoders which sensed the rotation of motor shafts and eye were recorded in real time. Since the encoder data was not a part of the control process, the measurement of the data was done simultaneously by another Arduino Uno. Two additional Arduino Unos were acquired; one to record readings from the right and left eye encoders and the other to record the encoder readings from the pan motor.

Figure 70 shows the example code for reading the voltage signal from the encoder. The voltage pulses were read from encoder channels A and B by using “pinMode”. They were first initialized as “HIGH” by using “digitalWrite” and the state of their voltage pulses were monitored by using interrupts. Since Arduino Uno has a small processer, only two interrupt pins could be used. The interrupt allowed the encoders to be monitored at all times rather than polling the voltage pulses in a loop. This not only sped up the processing time of the program but also ensured that all the readings were stored and monitored. The interrupt was declared by using “attachInterrupt” such that every time channel A of the left and right eye encoders changed voltage signals, the functions “doLeftEncoder” and “doRightEncoder” were called respectively.

```
digitalWrite (LeftEncoderA, HIGH); // turn on pull up resistor
pinMode(LeftEncoderA, INPUT);
digitalWrite (LeftEncoderB, HIGH); // turn on pull up resistor
pinMode(LeftEncoderB, INPUT);
pinMode(RightEncoderA, INPUT);
digitalWrite (RightEncoderA, HIGH); // turn on pull up resistor
pinMode(RightEncoderB, INPUT);
digitalWrite (RightEncoderB, HIGH); // turn on pull up resistor
attachInterrupt(0, doLeftEncoder, CHANGE);  // encoder left pin A on interrupt 0 - pin 2
attachInterrupt(1, doRightEncoder, CHANGE);  // encoder right pin A on interrupt 1 - pin 3
```

Figure 70: Example Code to Read Encoder Values

Figure 71 shows the example code to make counts of encoder readings so that the counts can be used to calculate rotation in degrees.

```
void doLeftEncoder() {
  if (digitalRead(LeftEncoderA) == digitalRead(LeftEncoderB)) {
    LeftValue++;
  } else {
    LeftValue--;
  }
}
```

Figure 71: Example Code to Make Counts of Encoder Readings

After the counts were performed, they needed to be updated as new and old count values in a timer loop so that they could be used for calculation of velocity and acceleration. The sampling time was chosen to be 1 millisecond through timer interrupt. Every 1 millisecond, timerIsr() function in Figure 72, which counted time, was called.

```
  /*----Make Timer ISR for sampling---*/
  Timer1.initialize(1000); // set a timer of length 1 millisecond.
  Timer1.attachInterrupt( timerIsr ); // attach the service routine here. This will act as the timer
}

void timerIsr()
{
  time++;
}
```

Figure 72: Example Code to Declare Timer Counts

When the count time changed, meaning a duration of 1 ms had passed, encounter count values were updated to new count values. Old and new count values were sent to Serial for data import into Excel through PuTTY (see Section 10.2.1). After sending the data, old count values were made equal to the latest new count value.

The complete code used for focus point position testing is included in Appendix F: Arduino Code for Eye Pan Velocity, Acceleration Testing.

10.2.3Results

The number of encoder data points for pan motor, right and left eyes gathered over 18 s of operation was around 700. Figure 73, Figure 74, Figure 75 and Figure 76 show pan motor velocity and acceleration, encoder values, eye velocity and acceleration respectively. It can be seen that the error is magnified as the data are differentiated. The error comes mostly from Arduino Uno timer count and the data transferred through PuTTY. It was found from the Excel data import that the data was not being transferred at the specified sample time of 1 ms. Instead, it was being transferred at a time interval of 18 ms to 22 ms. This caused some of the readings to be skipped and paired readings with incorrect time intervals. To fix those errors, data at constant time steps have to be generated by interpolation. After the data have been corrected for constant time step, the data can be filtered in MATLAB. This correction was not carried out due to time constraints.

From Figure 73, it can be seen that the maximum pan motor speed achieved was around 500 deg/s. Every motor has its own torque capacity at a specific angular velocity since

$$P = T \times \omega$$

where P = Power capacity of the motor

T = Output Torque

ω = Speed of the motor

As the motor speed is increased, the available torque is reduced. The speed was increased until the output torque of the motor is sufficient to overcome the friction in the mechanism.

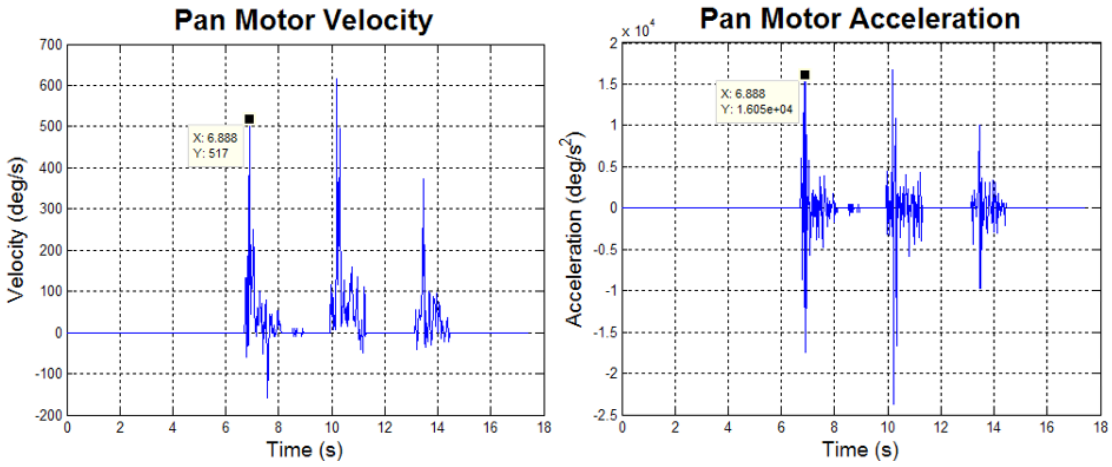


Figure 73: Calculated Velocity and Acceleration of Pan Motor

The encoder values plotted in Figure 74 show that the left (in red) and right (in blue) eyes moved at different increments to achieve coupled pan. The focus point moved from the origin at point A to the extreme left at B. Since the left eye had to pan more than the right eye in order for the eyes to stay focused on the same point, the encoder value at the left eye (in red) is greater than the encoder value at the right eye (in blue). On the other hand, when the focus point moved from the extreme left at B to the extreme right at C, the right eye had to pan more than the left eye and therefore the value of right eye at C is greater. During the path C to D, the focus point got back from the extreme right to the origin. The values stayed almost constant during the path DE and FG when the convergence depth was changed. This explains that the change of pan angles when the depth was changed is small as opposed to that when the eyes were panned. The change of pan angles was highest when the eyes are panned at the minimum convergence depth.

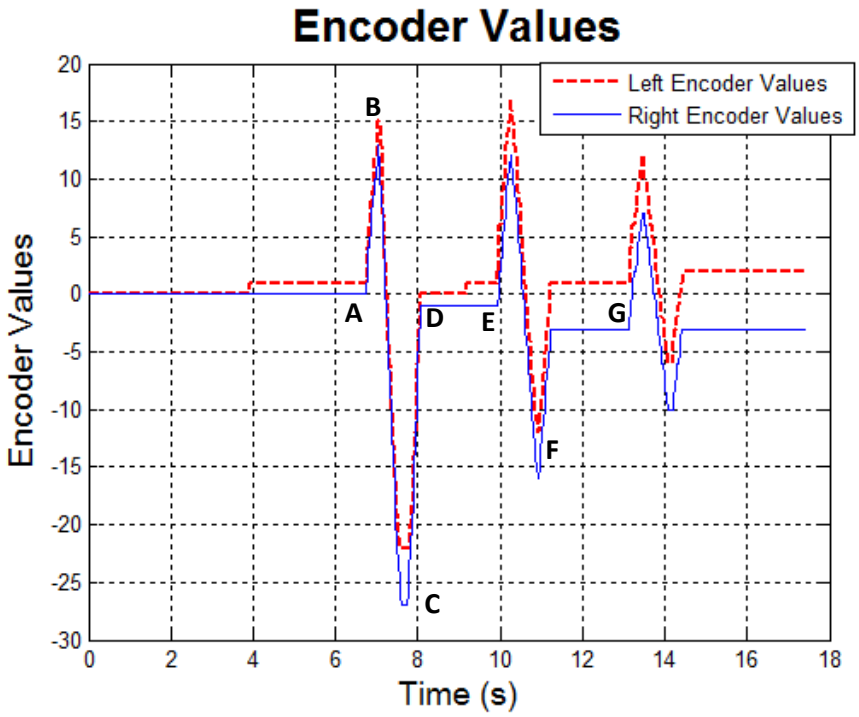


Figure 74: Encoder Values Mounted at Left and Right Eyes

This understanding agrees with velocity data graphs in Figure 75. The focus point moved from the origin to the extreme left during the path AB, from the extreme left to the extreme right during the path BC and from the extreme right to the origin during the path CD. The maximum eye velocity achieved was approximately 230.8 deg/s which is approximately 46% of human saccadic angular eye velocity of 500 deg/s.

The eye motor velocity depends on the motor output velocity limit. This limit was determined by the I^2C (Inter-Integrated Circuit) bus speed specified in Adafruit Stepper Motor Shield Library which controls the stepping rate of the stepper motor. After the bus speed in the stepper motor library was modified from an initial value of 100 kHz to 150 kHz, it was found that the stepper motor rotated at a faster speed but missed steps during rotation because the motor torque capacity at that particular speed was exceeded. Therefore, the eye angular velocity of 230.8 deg/s achieved from testing at the bus speed of 100 kHz was the maximum value while overcoming the torque required for rotation of all steps.

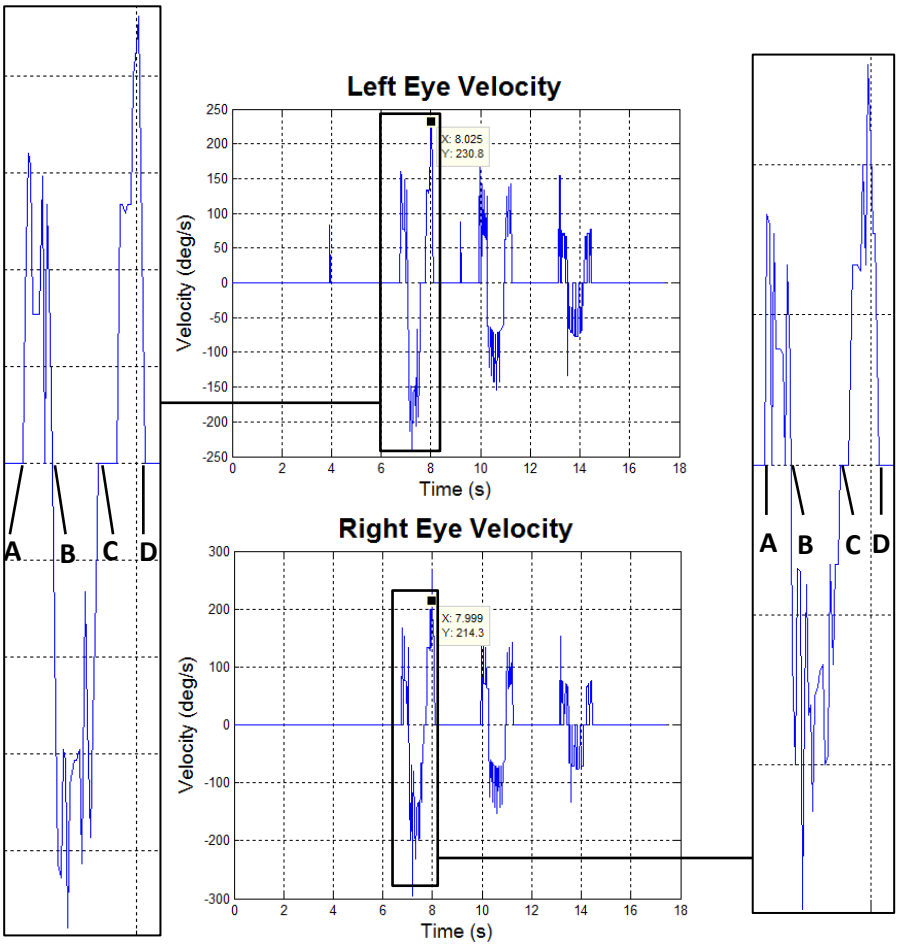


Figure 75: Velocity Calculated at Left Eye (Top) and Right Eye (Bottom)

It is hard to interpret from the acceleration data graphs for left and right eyes shown in Figure 76 due to noises and errors, and a reliable conclusion cannot be made. However, it can be seen that the maximum eye acceleration fell within the range of 4000 to 6000 deg/s^2. This range was about 16% to 24% of human eye acceleration during saccadic motion.

The angular acceleration of the eye was lower than expected due to more friction and resistance in the mechanism than was expected due to manufacturing and part mating imperfections. Additionally, the motor was not able to achieve the torque output specification because the power supply used was not able to produce the 24 V necessary, nor was the motor shield able to receive more than 12V. The reduced acceleration also contributed to the reduced velocity.

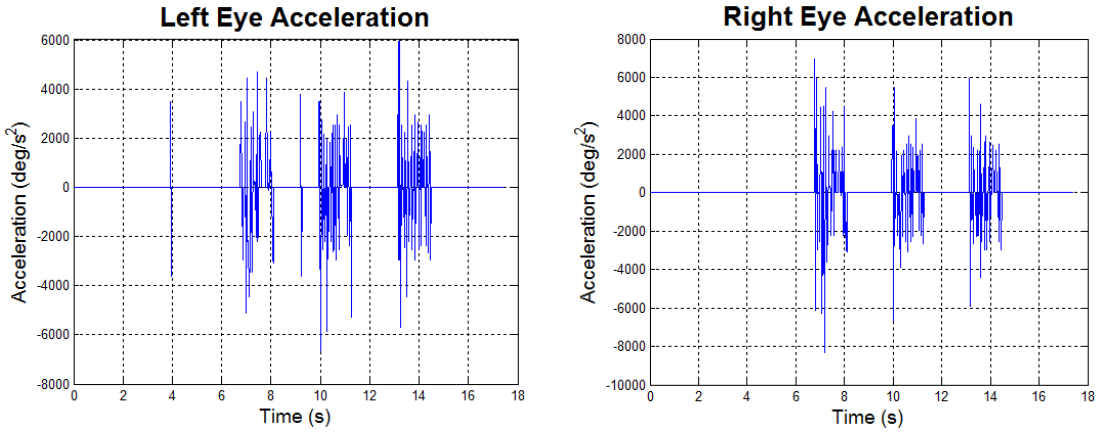


Figure 76: Acceleration Found at Left Eye (Left) and Right Eye (Right)

11 Conclusion

A mechanical platform capable of coupling pan motion, coupling vergence motion, and coupling tilt motion of a pair of eyes was designed, analyzed, prototyped and tested. This was done to investigate the potential of such a mechanism as a mechanical solution to the image processing correspondence problem.

A platform design that utilized linear actuation for pan and vergence motions and rotational actuation for tilt motions was selected for development. A mathematical model of this design was created and studied extensively. The platform was then constructed as a computer-generated model in Creo Parametric and as a physical prototype. Due to their availability and ease of use, Vex robotics parts obtained from the WPI Robotics Laboratory were largely incorporated into the prototype. Other components were purchased, laser-cut from acrylic, modified versions of standard parts, or created by a rapid prototyping machine.

The platform achieved coupling of each of the three degrees of freedom of the pair of eyes. Additionally, the computer-generated model of the platform and the results of the mechanical prototype tests validated the theoretical platform analysis by verifying that the actual focal point position of the eyes matched the theoretical position, within a range of error, for a given geometrical configuration of the platform driving mechanisms.

From the tests of the mechanical prototype, it was observed that this physical realization of the platform yielded inconsistent position error due to play and backlash between components, and fluctuations in velocity and acceleration due to uneven friction throughout the slider track surfaces. These errors arose from the relatively low quality of some of the components used and not from the platform design concept. The prototype demonstrated that mechanical coupling of the degrees of freedom of the eyes is a viable alternative solution to the image processing correspondence problem.

11.1 Recommendations

The following are recommended for future improvements of the mechanical platform:

1. *Mechanical parts with smoother surfaces and precise mating capability should be used to reduce friction, play, and backlash in links and joints.*
   Standard or custom parts made with high precision design and manufacturing would significantly reduce the issues of imprecise mating and uneven friction and, thus, unpredictable errors in the focal points of the eyes.
2. *More precise and compact actuation should be used to increase precision and accuracy of movement and reduce the size of the platform for practical mounting.*
   The relatively low quality of the sliders reduced the precision and accuracy of the translation of motion from the actuators to the platform eyes. In addition, the stepper motors and the sliders used for this prototype occupied a large amount of space relative to the other components of the platform.
3. *Cameras to sense laser focal point position error for platform calibration through closed loop control.*
   Before cameras are mounted in the eyes, the lasers can be used in conjunction with an external camera for initial platform calibration. The camera would be able to sense the position errors of the laser focal points. These errors would be used to determine values for modifying the input instructions in order to achieve the desired output positions.

4. *Serial Peripheral Interface (SPI) reading should be implemented in order to obtain absolute encoder values for platform calibration.*
   Time and resource limitations prevented the implementation of SPI reading, and thus allowed for only incremental encoder values to be read from the absolute encoders. Absolute encoder values would be useful for improved tracking of the platform configurations at all times, especially at start up.

Works Cited

Alexander, Elizabeth. “Affordable Compact Humanoid Robot for Autism Spectrum Disorder in Children.” Worcester Polytechnic Institute, 2011. Print.

Ali, Abdulrahman K. *Remote Sensing*. 2010. <http://www.uotechnology.edu.iq/appsciences/Laser/Lacture_laser/thrid_class/Remote_Sensing/3-Remote_Sensing.pdf>.

APC International, Ltd. *Piezoelectirc Ceramics: Prinicples and Applications*. 2nd. ed: APC International, Ltd., 2002. Print.

Breazeal, Cynthia, et al. “Active Vision for Sociable Robots.” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*. (2001) Ed. Print.

Breazeal, Cynthia. “Kismet, the Robot.” 2013. September 5. <http://www.ai.mit.edu/projects/sociable/baby-bits.html>.

Bush, S. “Eye Muscle Inspires Piezo Actuator.” (2006). <http://www.electronicsweekly.com/news/research/device-rd/eye-muscle-inspires-piezo-actuator-2012-07/>.

Goodrich, Michael A., et al. “A Case for Low-Dose Robotics in Autism Therapy.” *Proceedings of the 6th international conference on Human-robot interaction*. Ed. 1957702: ACM. Print.

Goodwin, S. “Robot Vision: Muscle-like Action Allows Camera to Mimic Human Eye Movement.” (2012) <http://www.news.gatech.edu/2012/07/05/robot-vision-muscle-action-allows-camera-mimic-human-eye-movement>.

Guizzo, Eric. “Superfast Robotic Camera Mimics Human Eye.” (2010). Print.

Hollerbach, John M., Ian W. Hunter, and John Ballantyne. “A Comparative Analysis of Actuator Technologies for Robotics.” (1992). <http://bdml.stanford.edu/twiki/pub/Main/PerchingLinks/Hollerbach_Hunter_Ballantyne_1992_A_Comparative_Analysis_of_Actuator_Technologies_for_Robotics.pdf>.

Iovine, John. “Artificial Pneumatic Muscles.” (2000). <http://go.galegroup.com/ps/i.do?id=GALE%7CA66450810&v=2.1&u=mlin_c_worpoly&it=r&p=AONE&sw=w&asid=6a99dffd757cfbf776962c411f321b67>.

Karnopp, D., Margolis, D. L., Rosenberg, R. C., Books24x7, & Ebrary Academic Complete. (2012). System dynamics modeling and simulation of mechatronic systems. pp. xii, 636 p. ill. 624 cm.).

Khorram, Siamak, et al. *Remote Sensing*. 2012. Print.

Kyto, Mikko, et al. “Method for Measuring Stereo Camera Depth Accuracy Based on Stereoscopic Vision.” 2011. *Aalto University School of Science and Technology*. <http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=730582>.

Stephens, Kristi. “Normal Neck Range of Motion.” 2013. Web. <http://www.livestrong.com/article/95456-normal-neck-range-motion/>.

Montgomery, Ted M. “The Extraocular Muscles.” 2013. *Anatomy, Physiology and Pathology of the Human Eye*. September 26 2013. <http://www.tedmontgomery.com/the_eye/>.

Murray, David, Fenglei Du, and Philip McLauchlan. “Design of Stereo Heads.” *Active Vision*. MIT Press, 1992. Print.

Orban de Xivry, Jean-Jacques, and Philippe Lefevre. “Saccades and Pursuit: Two Outcomes of a Single Sensorimotor Process.” 584.1 (2007): 11-23 pp.

"Personal Robots." MIT Media Lab, 2013. Massachusetts Institute of Technology. <http://media.mit.edu/research/groups/personal-robots>.

Quaia, Christian and Lance M. Optican. "Three-Dimensional Rotations of the Eye." *Adler's Physiology of the Eye.* 2001. Print.

Robotics; Humanoid Robot Helps Train Children with Autism. Atlanta: NewsRx, 2013. Print.

Schultz, Joshua, and Jun Ueda. "A Camera Positioner Driven by Muscle-Like Actuation." *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. Ed.: IEEE. Print.

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot modeling and control*. Hoboken, NJ: John Wiley & Sons.

Stephens, Kristi. "Normal Neck Range of Motion." 2013. <http://www.livestrong.com/article/95456-normal-neck-range-motion/>.

Taylor, K. "Robot Eye Moves Like the Real Thing." (2012). <http://www.tgdaily.com/trendwatch-features/64504-robot-eye-moves-like-the-real-thing>.

Thomas, Rob. (2009, December 1). Animatronic Eye Project UH SFX [Video file]. Retrieved from https://www.youtube.com/watch?v=lJ6McUUIb7Q

Ueda, Jun, Thomas Secord, and H. Harry Asada. *Large Effective-Strain Piezoelectric Actuators Using Nested Cellular Architecture with Exponential Strain Amplification Mechanisms*: Georgia Institute of Technology, 2010. Print.

"VEX Robotics." *VEX Robotics.* 2014. <http://www.vexrobotics.com/>.

Villgrattner, T., R. Zandery, and H. Ulbrich. "Modeling and Simulation of a Piezo-Driven Camera Orientation System." *IEEE International Conference on Mechatronics*. 2009. Print.

Wang, Xuan-yin, et al. *Design and Kinematic Analysis of a Novel Humanoid Robot Eye Using Pneumatic Artificial Muscles*, 2008. Print.

Wolfe, T. B., M. G. Faulkner, and J. Wolfaardt. "Development of a Shape Memory Alloy Actuator for a Robotic Eye Prosthesis." (2005). <http://iopscience.iop.org/0964-1726/14/4/035/pdf/0964-1726_14_4_035.pdf>.

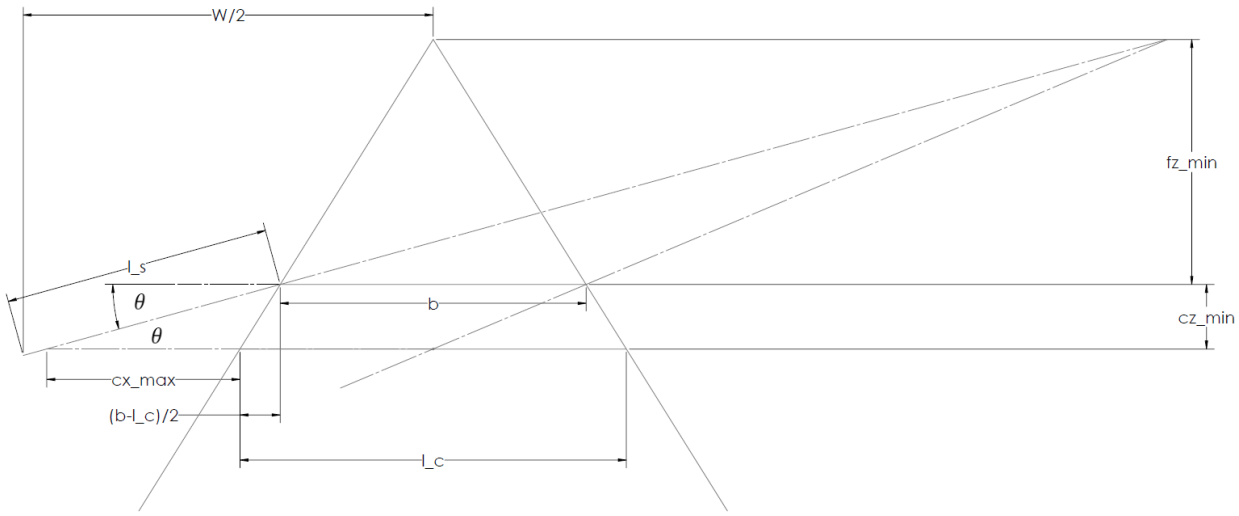
Appendices

Appendix A: Generating System of Equations to Determine Input and Output Design Parameters

Eleven equations were generated as follows. The focal point movement from the center of baseline is

$$fx_{max} = \frac{R}{2}$$

Equation 1



The above link configuration shows W which is the longest length the platform should accommodate.

$$W = 2 \times l_s \times \cos\theta + b$$

$$\theta = \tan^{-1}\frac{cz_{min}}{\frac{l_c - b}{2} + cx_{max}}$$

$$W = 2 \times l_s \times \cos\left(\tan^{-1}\frac{cz_{min}}{\frac{l_c - b}{2} + cx_{max}}\right) + b$$

Equation 2

The equations below were obtained from similarity triangle rule. The rule states that two triangles are similar when their corresponding angles are congruent and the corresponding sides have the same proportion.

$$\frac{fx_{max}}{cx_{max}} = \frac{fz_{max}}{cz_{max}}$$

Equation 3

$$\frac{fx_{max}}{cx_{max}} = \frac{fz_{min}}{cz_{min}}$$

Equation 4

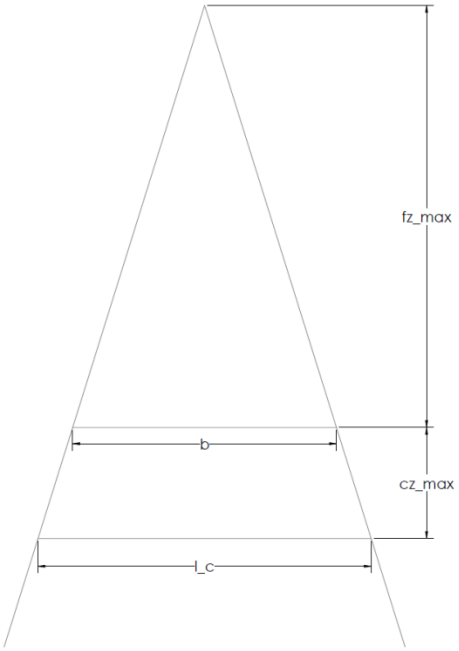
$$\frac{\Delta fx}{\Delta cx} = \frac{fz_{max}}{cz_{max}}$$

Equation 5

$$\frac{\Delta fz}{\Delta cz} = \frac{fz_{max}}{cz_{max}}$$

Equation 6

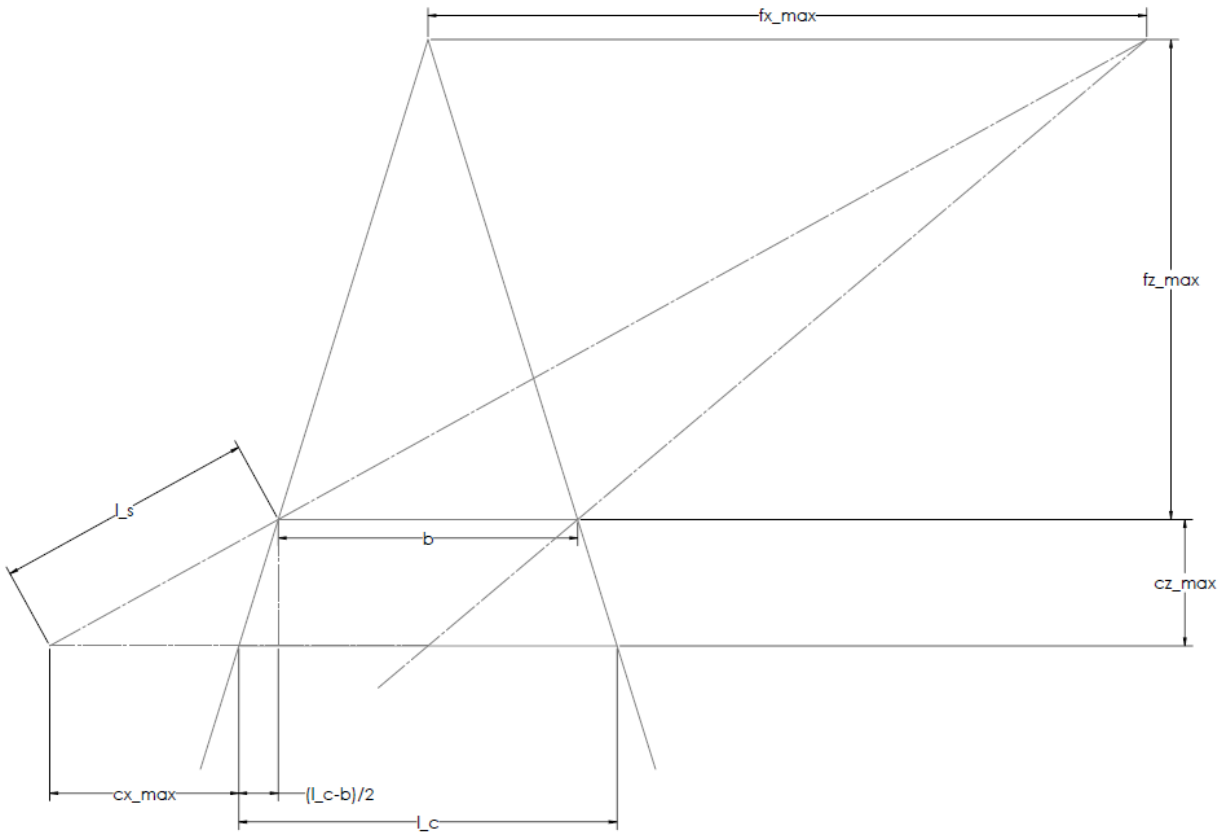
Note that fz_{max} and cz_{max} can also be represented in terms of b and l_c as shown in the figure below.



$$\frac{fz_{max}}{fz_{max} + cz_{max}} = \frac{b}{l_c}$$

Equation 7

To minimize the work space, it is assumed that the control bar is touching the end of the slider when the control bar is moved to left/right to see the focal point at the highest focal depth and the farthest left/right.



$$l_s{}^2 = cz_{max}{}^2 + (cz_{max} + \frac{l_c - b}{2})^2$$

Equation 8

The saccadic eye speed is pan angle of the focal point wrt the center of the baseline divided by time.

$$\omega_f = \frac{pan\ angle}{time}$$

$$pan\ angle = 2 \times \tan^{-1} \frac{fx_{max}}{fz_{min}}$$

$$time = \frac{2 \times fx_{max}}{v_f} = \frac{2 \times fx_{max}}{v_c \times \frac{fx_{max}}{cx_{max}}} = \frac{2 \times cx_{max}}{v_c} = \frac{2 \times cx_{max}}{r \times \omega_{motor} \times 2\pi}$$

Note: v=r ω is only true when $\omega$is in rad/s therefore $\omega_{motor}$is multiplied with 2π to change its unit from rev/s to rad/s .

Substituting pan angle and time in ω_f:

$$\omega_f = \frac{2 \times \tan^{-1} \frac{fx_{max}}{fz_{min}}}{\frac{2 \times cx_{max}}{r \times \omega_{motor} \times 2\pi}}$$

$$\omega_f = \frac{\tan^{-1}\frac{fx_{max}}{fz_{max}} \times r \times \omega_{motor} \times 2\pi}{cx_{max}}$$

Equation 9

The minimum increment the control bar moves (Δcx) is radius times the smallest angle the motor rotates in radians. Since the motor resolution is 200 steps/rev, 1 step is equal to $2\pi/200$ radian which is the angle in radian corresponding to Δcx.

$$\Delta cx = r_x \times \frac{2\pi}{200}$$

$$r_x = \frac{\Delta cx \times 200}{2\pi}$$

To represent it in general form, the radius of the pinion for control bar x movement is

$$r_x = \frac{\Delta cx \times motor_{res}}{2\pi}$$

Equation 10

Similarly, the radius of the pinion for control bar z movement is

$$r_z = \frac{\Delta cx \times motor_{res}}{2\pi}$$

Equation 11

Appendix B: Mathcad Initial Design Output Parameters Calculation

Input Design Parameters

Enter input values:

$b := 12.5$	cm	Eye baseline
$fz_{max} := 250$	cm	Max. focal depth from the baseline
$fz_{min} := 50$	cm	Min. focal depth from the baseline
$\Delta fx := 2$	cm	Focal point x resolution
$\Delta fz := 5$	cm	Focal point z resolution
$\omega_f := 500$	cm	Saccadic eye speed (for panning)
$motor_{res} := 200$	steps/rev	Stepper motor (for control bar x movement) resolution
$fx_{max} := 125$	cm	Maximum range of vision
$cz_{min} := 2$	cm	Min control bar z movement from the baseline

(can also choose one of the following cz.min,cz.max, deltacz or radius of pinion)

Output Design Parameters

Enter guess values of output:

Guess		
$W := 40$	cm	Max distance of the workspace
$l_c := 13$	cm	Control bar length
$l_s := 15$	cm	Slider length
$cz_{max} := 13$	cm	Max control bar z movement from the baseline
$\Delta cx := 0.1$	cm	Min. x increment of the control bar corresponding to x focal resolution
$\Delta cz := 0.25$	cm	Min. z increment of the control bar corresponding to z focal resolution
$r_x := 2$	cm	Radius of the pinion for rack x movement
$r_z := 5$	cm	Radius of the pinion for rack z movement
$cx_{max} := 6$	cm	Max control bar x movement to get the focal point from the center of the baseline to the farthest right/left
$\omega_{motor} := 3$	rev/s	Speed of the pan stepper motor (for x control bar movment)

System of Equations:

Given

$$W = 2 \cdot l_s \cdot \cos\left(\operatorname{atan2}\left(\frac{l_c - b}{2} + cx_{max}, cz_{min}\right)\right) + b$$

$$\frac{fx_{max}}{cx_{max}} = \frac{fz_{max}}{cz_{max}}$$

$$\frac{fz_{max}}{cz_{max}} = \frac{fz_{min}}{cz_{min}}$$

$$\frac{fz_{max}}{cz_{max}} = \frac{b}{l_c - b}$$

$$l_s^{\,2} = cz_{max}^{\,2} + \left(\frac{l_c - b}{2} + cx_{max}\right)^2$$

$$\frac{\Delta fx}{\Delta cx} = \frac{fz_{max}}{cz_{max}}$$

$$\frac{\Delta fz}{\Delta cz} = \frac{fz_{max}}{cz_{max}}$$

$$\omega_f = \operatorname{atan2}(fz_{max}, fx_{max}) \cdot \frac{180}{\pi} \cdot \omega_{motor} \cdot r_x \cdot \frac{2 \cdot \pi}{cx_{max}}$$

$$r_x = \Delta cx \cdot \frac{motor_{res}}{2\pi}$$

$$r_z = \Delta cz \cdot \frac{motor_{res}}{2 \cdot \pi}$$

Results:

$$\operatorname{Find}(W, l_c, l_s, cz_{max}, cx_{max}, \Delta cx, \Delta cz, \omega_{motor}, r_x, r_z) \rightarrow \begin{pmatrix} 33.608876177223327837 & -8.6088761772233278372 \\ 13.0 & 13.0 \\ 11.294356998076517326 & -11.294356998076517326 \\ 10.0 & 10.0 \\ 5.0 & 5.0 \\ 0.08 & 0.08 \\ 0.2 & 0.2 \\ 5.8817880288829410511 & 5.8817880288829410511 \\ 2.5464790894703253723 & 2.5464790894703253723 \\ 6.3661977236758134308 & 6.3661977236758134308 \end{pmatrix}$$

Appendix C: Mathcad Final Design Parameters Calculation

Input Design Parameters

Enter input values:

$b := 12.5$	cm	Eye baseline
$cz_{max} := 13.1$	cm	Max control bar z movement from the baseline
$cx_{max} := 5$	cm	Max control bar movement to get the focal point from the center of the baseline to the farthest right/left
$r_x := \frac{3.81}{2} = 1.905$	cm	Pan motor pinion radius
$r_z := \frac{3.81}{2} = 1.905$	cm	Convergence motor pinion radius
$\omega_f := 500$	rev/s	Saccadic eye speed (You can also choose w.motor)
$motor_{res} := 200$	steps/rev	Stepper motor for pan and convergence resolution
$l_c := 13.1$	cm	Control bar length
$cz_{min} := 3.6$	cm	Min control bar z movement from the baseline

Output Design Parameters

Enter guess values of output:

Guess

$W := 40$	cm	Max distance of the workspace
$fx_{max} := 100$	cm	Max range of vision
$l_s := 15$	cm	Slider Length
$fz_{min} := 50$	cm	Min. focal depth from the baseline
$fz_{max} := 250$	cm	Max. focal depth from the baseline
$\Delta cx := 0.1$	cm	Min. x increment of the control bar corresponding to x focal resolution
$\Delta cz := 0.25$	cm	Min. increment of the control bar corresponding to z focal resolution
$\Delta fx := 2$	cm	Focal point x resolution
$\Delta fz := 2$	cm	Focal point z resolution
$\omega_{motor} := 10$	rev/s	Maximum angular speed of the pan stepper motor (for control bar x movement)

System of Equations:

Given

$$W = 2 \cdot l_s \cdot \cos\left(\operatorname{atan2}\left(\frac{l_c - b}{2} + cx_{max}, cz_{min}\right)\right) + b$$

$$\frac{fx_{max}}{cx_{max}} = \frac{fz_{max}}{cz_{max}}$$

$$\frac{fz_{max}}{cz_{max}} = \frac{fz_{min}}{cz_{min}}$$

$$\frac{fz_{max}}{cz_{max}} = \frac{b}{l_c - b}$$

$$l_s^{\,2} = cz_{max}^{\,2} + \left(\frac{l_c - b}{2} + cx_{max}\right)^2$$

$$\frac{\Delta fx}{\Delta cx} = \frac{fz_{max}}{cz_{max}}$$

$$\frac{\Delta fz}{\Delta cz} = \frac{fz_{max}}{cz_{max}}$$

$$\omega_f = \operatorname{atan2}(fz_{max}, fx_{max}) \cdot \frac{180}{\pi} \cdot \omega_{motor} \cdot r_x \cdot \frac{2 \cdot \pi}{cx_{max}}$$

$r_x = \Delta cx \cdot \frac{motor_{res}}{2\pi}$ The radius of the pinion for lateral control bar movement

$r_z = \Delta cz \cdot \frac{motor_{res}}{2 \cdot \pi}$ The radius of the pinon for proximal control bar movement

Results:

$$\operatorname{Find}(W, fx_{max}, l_s, fz_{min}, fz_{max}, \Delta cx, \Delta cz, \Delta fx, \Delta fz, \omega_{motor}) \rightarrow \begin{pmatrix} -10.879666775084868779 & 35.879666775084868779 \\ 104.16666666666666666 & 104.16666666666666666 \\ -14.131525041551601765 & 14.131525041551601765 \\ 74.999999999999999998 & 74.999999999999999998 \\ 272.91666666666666666 & 272.91666666666666666 \\ 0.059847340050885561192 & 0.059847340050885561192 \\ 0.059847340050885561192 & 0.059847340050885561192 \\ 1.2468195843934491915 & 1.2468195843934491915 \\ 1.2468195843934491915 & 1.2468195843934491915 \\ 9.9979180855324777492 & 9.9979180855324777492 \end{pmatrix}$$

Appendix D: Mathcad Torque and Speed Analysis

INPUT:

Insert the desired max velocity and max acceleration
Our initial design specifications were 500 deg/sec and 25,000 deg/sec^2

$$max\_vel := 400 \frac{deg}{s}$$

$$max\_acc := 20000 \frac{deg}{s^2}$$

$$spec\_max\_vel := 500 \frac{deg}{s}$$

$$spec\_max\_acc := 25000 \frac{deg}{s^2}$$

Insert the minimum range of movement of the eye from one extreme to the other,
and the corresponding range of movement of the control bar

$$eye\_range := 40deg$$

$$bar\_range := 10cm$$

Insert the amount of mass being moved by the slider and the radius of the pinion

$$mass := .2543kg$$

$$pinion\_r := 1.905cm$$

Other:

$\mu_s := 0.3$ Coefficient of static friction between delrin and steel

EQUATIONS:

$$t_a(vel, acc) := \frac{vel}{acc}$$

Time it takes for the eye to acclerate to the max velocity, which is also the time it takes to deccelerate to 0 velocity

$$x_a(acc, t_a) := \frac{1}{2} \cdot acc \cdot (t_a)^2$$

Distance moved by the eye during accleration, which is also the distance covered during decceleration

$$x_c(x_a) := eye\_range - 2 \cdot x_a$$

Distance moved by the eye during period of constant velocity

$$t_c(x_c, vel) := \frac{x_c}{vel}$$

Time it takes for the eye to cover the distance during constant velocity

$$v_{bar}(x_c, t_c) := \frac{\frac{x_c}{eye\_range} \cdot bar\_range}{t_c}$$

Maximum velocity attained by the control bar

$$a_{bar}(v_{bar}, t_a) := \frac{v_{bar}}{t_a}$$

Maximum acceleration attained by the control bar

$$Torque_a(a_{bar}) := a_{bar} \cdot mass \cdot pinion\_r$$

Motor torque necessary to achieve acceleration

$$Torque_\mu := mass \cdot g \cdot \mu_s \cdot pinion\_r$$

Motor torque necessary to overcome static friction

$$Speed(v_{bar}) := \frac{v_{bar}}{pinion\_r} \cdot \frac{1}{2 \cdot \pi} \cdot 200$$

Required motor speed in pps

CALCULATIONS:

for inputted velocity and acceleration:

$t_{a\_in} := t_a(max\_vel, max\_acc) = 0.02\,s$ Eye accleration time

$x_{a\_in} := x_a(max\_acc, t_{a\_in}) = 4 \cdot deg$ Distance moved during accleration

$x_{c\_in} := x_c(x_{a\_in}) = 32 \cdot deg$ Distance moved by the eye with constant velocity

$t_{c\_in} := t_c(x_{c\_in}, max\_vel) = 0.08\,s$ Duration of eye travel during constant velocity

$v_{bar\_in} := v_{bar}(x_{c\_in}, t_{c\_in}) = 1\,\frac{m}{s}$ Max control bar velocity

$a_{bar\_in} := a_{bar}(v_{bar\_in}, t_{a\_in}) = 50\,\frac{m}{s^2}$ Max control bar acceleration

$Torque_{a\_in} := Torque_a(a_{bar\_in}) = 24.222 \cdot N \cdot cm$ Motor torque necessary to achieve acceleration

for specifications velocity and acceleration:

$t_{a\_spec} := t_a(spec\_max\_vel, spec\_max\_acc) = 0.02\,s$ Eye accleration time

$x_{a\_spec} := x_a(spec\_max\_acc, t_{a\_spec}) = 5 \cdot deg$ Distance moved during accleration

$x_{c\_spec} := x_c(x_{a\_spec}) = 30 \cdot deg$ Distance moved by the eye during constant velocity

$t_{c\_spec} := t_c(x_{c\_spec}, spec\_max\_vel) = 0.06\,s$ Duration of eye travel with constant velocity

$v_{bar\_spec} := v_{bar}(x_{c\_spec}, t_{c\_spec}) = 1.25\,\frac{m}{s}$ Max control bar velocity

$a_{bar\_spec} := a_{bar}(v_{bar\_spec}, t_{a\_spec}) = 62.5\,\frac{m}{s^2}$ Max control bar acceleration

$Torque_{a\_spec} := Torque_a(a_{bar\_spec}) = 30.278 \cdot N \cdot cm$ Motor torque necessary to achieve acceleration

OUTPUT:

for inputted velocity and acceleration:

$$Torque_{Ncm\_in} := Torque_{a\_in} + Torque_{\mu} = 25.647 \cdot N \cdot cm$$

Total motor torque required in Ncm for the inputted acceleration

$$Torque_{gcm\_in} := Torque_{Ncm\_in} \cdot \frac{1000 \frac{g}{kg}}{g} = 2.615 \times 10^{3} \cdot g \cdot cm$$

Total motor torque required in gcm for the inputted acceleration

$$Speed\_in := Speed\left(v_{bar\_in}\right) = 1.671 \times 10^{3} \frac{1}{s}$$

Required motor speed in pps for the inputted speed

for specifications velocity and acceleration:

$$Torque_{Ncm\_spec} := Torque_{a\_spec} + Torque_{\mu} = 31.703 \cdot N \cdot cm$$

Total motor torque required in Ncm for the specification acceleration

$$Torque_{gcm\_spec} := Torque_{Ncm\_spec} \cdot \frac{1000 \frac{g}{kg}}{g} = 3.233 \times 10^{3} \cdot g \cdot cm$$

Total motor torque required in gcm for the specification acceleration

$$Speed\_spec := Speed\left(v_{bar\_spec}\right) = 2.089 \times 10^{3} \frac{1}{s}$$

Required motor speed in pps for the specification speed

Appendix E: Arduino Code for Focus Point Position Testing

```
// Motor Eyes MQP
// This program tests position of the focal point by open loop control of steps, max speed and acceleration.
//
// Requires the Adafruit_Motorshield v2 library
//   https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library
// And AccelStepper with AFMotor support
//   https://github.com/adafruit/AccelStepper

#include <AccelStepper.h>
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <math.h>

/*******************DECLARE/ASSIGN PARAMETERS*****************************/
/*------Stepper Motors------*/
Adafruit_MotorShield AFMSbot(0x61);        // Rightmost jumper closed
Adafruit_MotorShield AFMStop(0x60);        // Default address, no jumpers
// Connect two steppers with 200 steps per revolution (1.8 degree)
// to the top shield
Adafruit_StepperMotor *myPanStepper = AFMStop.getStepper(200, 1);
Adafruit_StepperMotor *myConStepper = AFMStop.getStepper(200, 2);
// Connect one stepper with 200 steps per revolution (1.8 degree)
// to the bottom shield
Adafruit_StepperMotor *myTiltStepper = AFMSbot.getStepper(200, 1);
// Can change these to DOUBLE or INTERLEAVE or MICROSTEP!
// Wrappers for the first motor!
void forwardstep1() {
  myPanStepper->onestep(FORWARD, SINGLE);
}
void backwardstep1() {
  myPanStepper->onestep(BACKWARD, SINGLE);
}
// wrappers for the second motor!
void forwardstep2() {
  myConStepper->onestep(FORWARD, SINGLE);
}
void backwardstep2() {
  myConStepper->onestep(BACKWARD, SINGLE);
}
// wrappers for the third motor!
void forwardstep3() {
  myTiltStepper->onestep(FORWARD, SINGLE);
}
void backwardstep3() {
  myTiltStepper->onestep(BACKWARD, SINGLE);
}
// Wrap the 3 steppers in an AccelStepper object
AccelStepper PanStepper(forwardstep1, backwardstep1);
AccelStepper ConStepper(forwardstep2, backwardstep2);
AccelStepper TiltStepper(forwardstep3, backwardstep3);

/*------Stepper Motors------*/
int X[3]={
  0,86,-86};
int Y[3]={
  0,-25,25};
int Z[3]={
  0,95,190};

int buttonPin = 6;
```

```
/*******************INITIALIZE THE WHOLE PROGRAM*****************************/
void setup()
{
  pinMode(buttonPin, INPUT);

  /*----Initialize Motor Speeds---*/
  AFMSbot.begin();         // Start the bottom shield, create with the default frequency 1.6 kHz
  AFMStop.begin();         // Start the top shield, create with the default freq

  PanStepper.setMaxSpeed(80.0);
  PanStepper.setAcceleration(200.0);

  ConStepper.setMaxSpeed(80.0);
  ConStepper.setAcceleration(200.0);

  TiltStepper.setMaxSpeed(0.0);
  TiltStepper.setAcceleration(70.0);

  Serial.begin (9600);
  Serial.println("Start");
}
```

Appendix F: Arduino Code for Eye Pan Velocity, Acceleration Testing

Motor Control

```
// Motor Eyes MQP
// This program tests max speed and acceleration of the eyes.
//
// Requires the Adafruit_Motorshield v2 library
//   https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library
// And AccelStepper with AFMotor support
//   https://github.com/adafruit/AccelStepper

// This tutorial is for Adafruit Motorshield v2 only!
// Will not work with v1 shields

#include <AccelStepper.h>
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <TimerOne.h>

/*******************DECLARE/ASSIGN PARAMETERS*****************************/
/*------Stepper Motors------*/
Adafruit_MotorShield AFMSbot(0x61);        // Rightmost jumper closed
Adafruit_MotorShield AFMStop(0x60);        // Default address, no jumpers
// Connect two steppers with 200 steps per revolution (1.8 degree)
// to the top shield
Adafruit_StepperMotor *myPanStepper = AFMStop.getStepper(200, 1);
Adafruit_StepperMotor *myConStepper = AFMStop.getStepper(200, 2);
// Connect one stepper with 200 steps per revolution (1.8 degree)
// to the bottom shield
Adafruit_StepperMotor *myTiltStepper = AFMSbot.getStepper(200, 2);
// Can change these to DOUBLE or INTERLEAVE or MICROSTEP!
// Wrappers for the first motor!
void forwardstep1() {
  myPanStepper->onestep(FORWARD, SINGLE);
}
void backwardstep1() {
  myPanStepper->onestep(BACKWARD, SINGLE);
}
// wrappers for the second motor!
void forwardstep2() {
  myConStepper->onestep(FORWARD, SINGLE);
}
void backwardstep2() {
  myConStepper->onestep(BACKWARD, SINGLE);
}
// wrappers for the third motor!
void forwardstep3() {
  myTiltStepper->onestep(FORWARD, SINGLE);
}
void backwardstep3() {
  myTiltStepper->onestep(BACKWARD, SINGLE);
}
// Wrap the 3 steppers in an AccelStepper object
AccelStepper PanStepper(forwardstep1, backwardstep1);
AccelStepper ConStepper(forwardstep2, backwardstep2);
AccelStepper TiltStepper(forwardstep3, backwardstep3);

int c[3]={0,95,190};

int switchcase=0;
int i,j=0;
```

```
/********************INITIALIZE THE WHOLE PROGRAM*****************************/
void setup()
{

  /*----Give Speed and Acceleration inputs at the motors---*/
  AFMSbot.begin();         // Start the bottom shield, create with the default frequency 1.6 kHz
  AFMStop.begin();         // Start the top shield, create with the default freq
  //AFMS.begin(1000);      // OR with a different frequency, say 1KHz

  TiltStepper.setMaxSpeed(0);

  PanStepper.setMaxSpeed(500);

  ConStepper.setMaxSpeed(-50);

  Serial.begin (9600);
}

/********************RUN THE WHOLE PROGRAM*****************************/
void loop()
{
while (i==0){
      PanStepper.moveTo(86);    // write the steps
      PanStepper.setSpeed(500.0);
      PanStepper.runSpeed();
      if(PanStepper.currentPosition()==86)
      i=1;
    }

while (i==1){
      PanStepper.moveTo(-86);    // write the steps
      PanStepper.setSpeed(-500.0);
      PanStepper.runSpeed();
      if(PanStepper.currentPosition()==-86)
        i=2;
      }

while (i==2){
      PanStepper.moveTo(0);    // write the steps
      PanStepper.setSpeed(500.0);
      PanStepper.runSpeed();
      if(PanStepper.currentPosition()==0){
        i=3;
        if (j<2)
        j++;
        else
        i=4;
      }
    }
while (i==3){
      ConStepper.moveTo(c[j]);
      ConStepper.setSpeed(-50.0);
      ConStepper.runSpeed();

      if(ConStepper.currentPosition()==c[j])
      i=0;
    }
while (i==4){
      PanStepper.stop();
      ConStepper.stop();
      TiltStepper.stop();
      Serial.write(100);
    }
}
```

Data Measurement of Encoders

```
// Motor Eyes MQP
// This program records the position of the eye encoder in a timer loop.
//
// Requires the Adafruit_Motorshield v2 library
//   https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library
// And AccelStepper with AFMotor support
//   https://github.com/adafruit/AccelStepper

// This tutorial is for Adafruit Motorshield v2 only!
// Will not work with v1 shields

#include <AccelStepper.h>
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"
#include <TimerOne.h>

/*******************DECLARE/ASSIGN PARAMETERS*****************************/
volatile long OldLeftValue=0;
volatile long OldRightValue=0;
volatile long NewLeftValue;
volatile long NewRightValue;
volatile long LeftValue=0;
volatile long RightValue=0;

volatile unsigned long time = 0;
volatile unsigned long newtime = 0;
volatile unsigned long lasttime = 0;


/*------Encoders------*/
const int LeftEncoderA = 2;              // pin that the encoder for left eye is attached to
const int LeftEncoderB = 4;
const int RightEncoderA = 3;              // pin that the encoder for right eye is attached to
const int RightEncoderB = 5;


/*******************INITIALIZE THE WHOLE PROGRAM*****************************/
void setup()
{
  /*----Attach interrupts to encoder---*/
  digitalWrite (LeftEncoderA, HIGH); // turn on pull up resistor
  pinMode(LeftEncoderA, INPUT);
  digitalWrite (LeftEncoderB, HIGH); // turn on pull up resistor
  pinMode(LeftEncoderB, INPUT);
  pinMode(RightEncoderA, INPUT);
  digitalWrite (RightEncoderA, HIGH); // turn on pull up resistor
  pinMode(RightEncoderB, INPUT);
  digitalWrite (RightEncoderB, HIGH); // turn on pull up resistor
  attachInterrupt(0, doLeftEncoder, CHANGE);  // encoder left pin A on interrupt 0 - pin 2
  attachInterrupt(1, doRightEncoder, CHANGE);  // encoder right pin A on interrupt 1 - pin 3
```

```
  /*----Print titles into Serials ---*/
  Serial.begin (9600);
  Serial.print("Last Time");
  Serial.print(",");
  Serial.print("New Time");
  Serial.print(",");
  Serial.print("Old Left Value");
  Serial.print(",");
  Serial.print("New Left Value");
  Serial.print(",");
  Serial.print("Old Right Value");
  Serial.print(",");
  Serial.print("New Right Value");
  Serial.print(",");
  Serial.print("Motor Pan Velocity");
  Serial.print(",");
  Serial.print("Motor Con Velocity");
  Serial.print(",");
  Serial.print("Old Left Velocity");
  Serial.print(",");
  Serial.print("Left Velocity");
  Serial.print(",");
  Serial.print("Old Right Velocity");
  Serial.print(",");
  Serial.print("Right Velocity");
  Serial.print(",");
  Serial.print("Left Acceleration");
  Serial.print(",");
  Serial.println("Right Acceleration");

  /*----Make Timer ISR for sampling---*/
  Timer1.initialize(1000); // set a timer of length 1 millisecond.
  Timer1.attachInterrupt( timerIsr ); // attach the service routine here. This will act as the timer
}

void timerIsr()
{
  time++;
}
```

```
/*******************RUN THE WHOLE PROGRAM*****************************/
void loop()
{

  /*----Acquire old,new timer and encoder values---*/
  if (time != lasttime){
    newtime = time;

    //Update encoder readings at left and right eyes
    NewLeftValue = LeftValue;
    NewRightValue = RightValue;

    //Send old,new timer and encoder values by printing into serial
    Serial.print(lasttime);
    Serial.print(",");
    Serial.print(newtime);
    Serial.print(",");
    Serial.print(OldPanValue);
    Serial.print(",");
    Serial.print(NewConValue);
    Serial.print(",");
    Serial.print(OldLeftValue);
    Serial.print(",");
    Serial.print(NewLeftValue);
    Serial.print(",");
    Serial.print(OldRightValue);
    Serial.print(",");
    Serial.println(NewRightValue);

    //Record old encoder values
    OldLeftValue = NewLeftValue;
    OldRightValue = NewRightValue;

    //Record old times
    lasttime = newtime;
}
}

void doLeftEncoder() {
  if (digitalRead(LeftEncoderA) == digitalRead(LeftEncoderB)) {
    LeftValue++;
  } else {
    LeftValue--;
  }
}

void doRightEncoder() {
  if (digitalRead(RightEncoderA) == digitalRead(RightEncoderB)) {
    RightValue++;
  } else {
    RightValue--;
  }
}
```

Appendix G: Derivation of Equations for Calculating Focus Point Position for Position Error Testing

The equation for calculating the actual tilt angle of the focus point is a trigonometric relation for the triangle formed by the line of sight of the eyes projected on the y-z plane, the known distance *a* from the eyes to the whiteboard, and the distance *h* – a manual measurement from the laser points on the board to the x-z (zero tilt) plane (see Figure 77):

$$\theta_{tilt,actual} = \tan^{-1}\left(\frac{h}{a}\right) * \frac{180}{\pi}$$

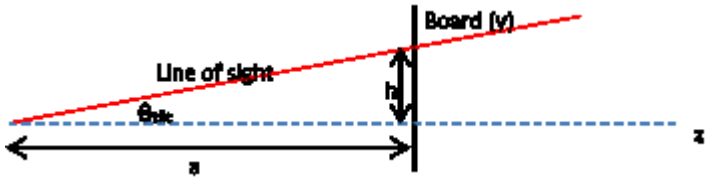


Figure 77. Diagram of Parameters for Tilt Angle Calculation

The equation for the actual convergence depth of the focus point, *fz*, is derived from the similar triangles shown highlighted in orange in Figure 78. For these cases, the larger triangle with height *fz* and base *b* is similar to the smaller triangle with height (*fz* – *a*) and base (*Rx* – *Lx*) with the appropriate sign convention used (Lx and Rx are positive to the right of the z-axis). Therefore:

$$\frac{b}{fz_{actual}} = \frac{Rx - Lx}{fz - a}$$

Rearranging,

$$fz_{actual} = \frac{b * a}{(b - Rx + Lx)}$$

For calculating actual *fx*, the similar triangles used to derive the equation is shown in Figure 78 highlighted in green. Here, the larger triangle with height *fz* (previously calculated) and base *fx* is similar to smaller triangle with height *a* and base equal to the distance from the x-z axes origin and the midpoint between the left and right laser points, *m*:

$$\frac{m}{a} = \frac{fx_{actual}}{fz_{actual}}$$

The midpoint distance *m* can be written in terms of the previously measured *Lx* and *Rx* as

$$m = \frac{Lx + Rx}{2}$$

Substituting into the *fx* equation,

$$\frac{Lx + Rx}{2 * a} = \frac{fx_{actual}}{fz_{actual}}$$

Rearranging,

$$fx_{actual} = \frac{fz * (Lx + Rx)}{2 * a}$$

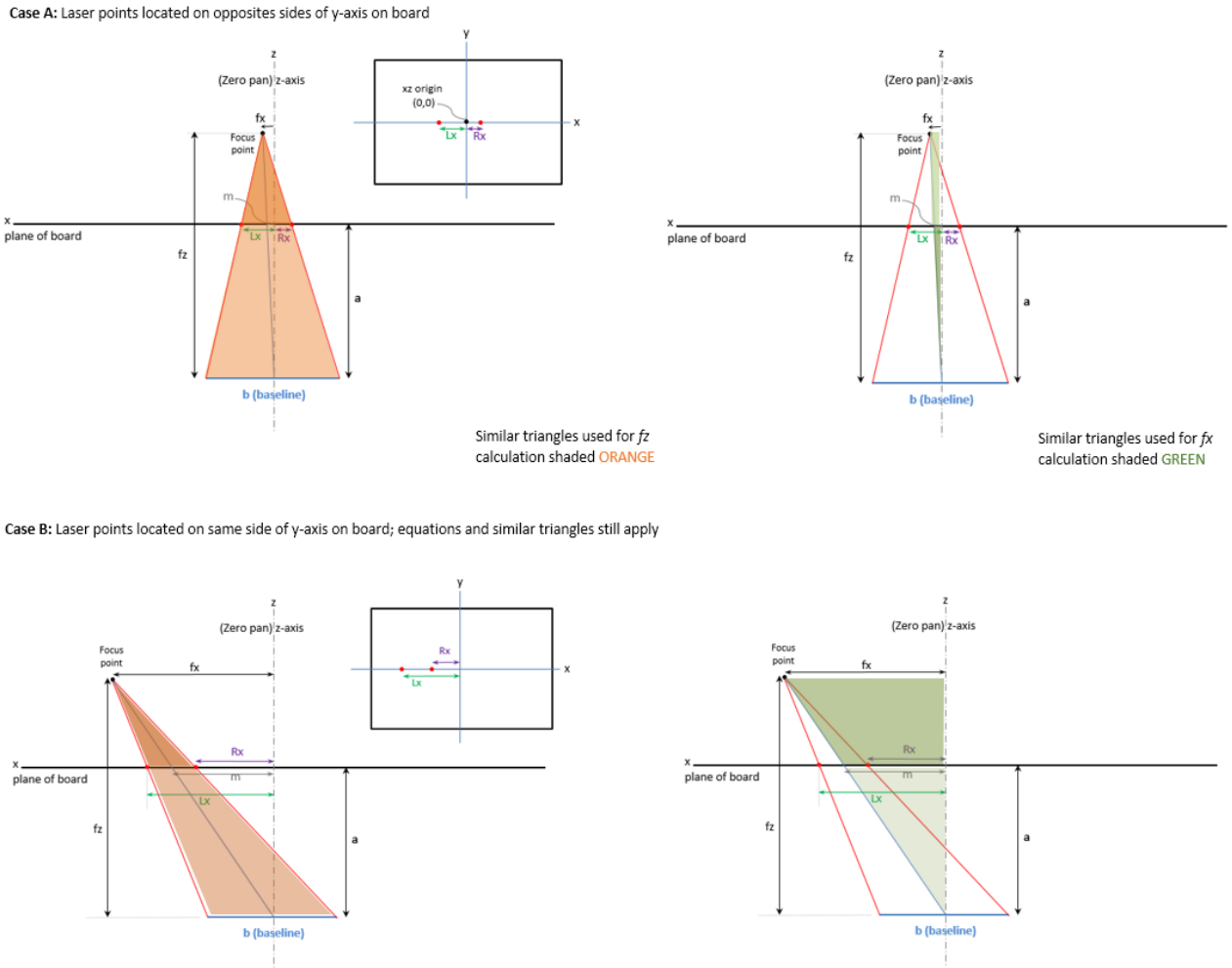
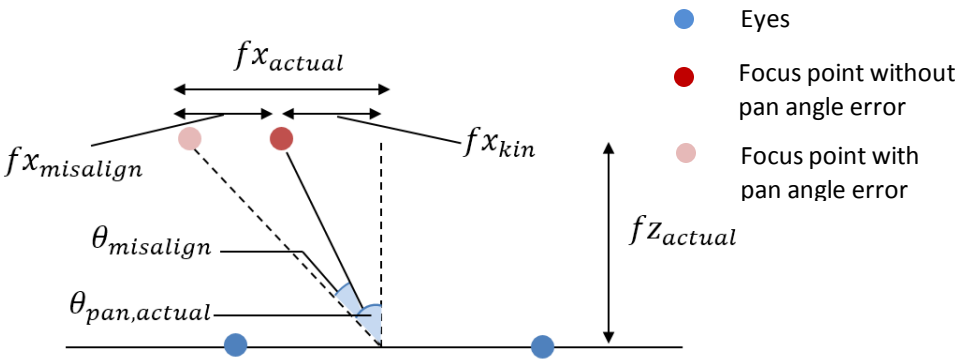


Figure 78. Diagram of similar triangles used to derive equations for actual fx (green) and fz (orange)

Appendix H: Position Calibration Analysis



INPUTS

$fx_{actual} := 53.9$ Experimental fx, cm

$b := 12.5$ Baseline, cm

$cx := 5$ Corresponding control bar position in x for given data point, cm

$fz_{actual} := 164.1$ Experimental fz, cm

$\theta_{panactual} := 18.189deg$ Experimental pan angle

$cz_{actual} := 13.1$ Corresponding control bar position in z for given data point, cm

Guess

$fx_{kin} := 104$ The resulting fx position for a given set of kinematic dimensions (b, cx, l.c),cm

$\theta_{misalign} := 2deg$ Effective angular misalignment of combined line of sight (extending from midpoint of eyes to focus point)

$fx_{misalign} := 5$ fx due to angular misalignment, cm

$l_c := 13.3$ Control bar length, cm

SYSTEM OF EQUATIONS

Given

$$fx_{actual} = fx_{kin} + fx_{misalign}$$ refer to the above figure

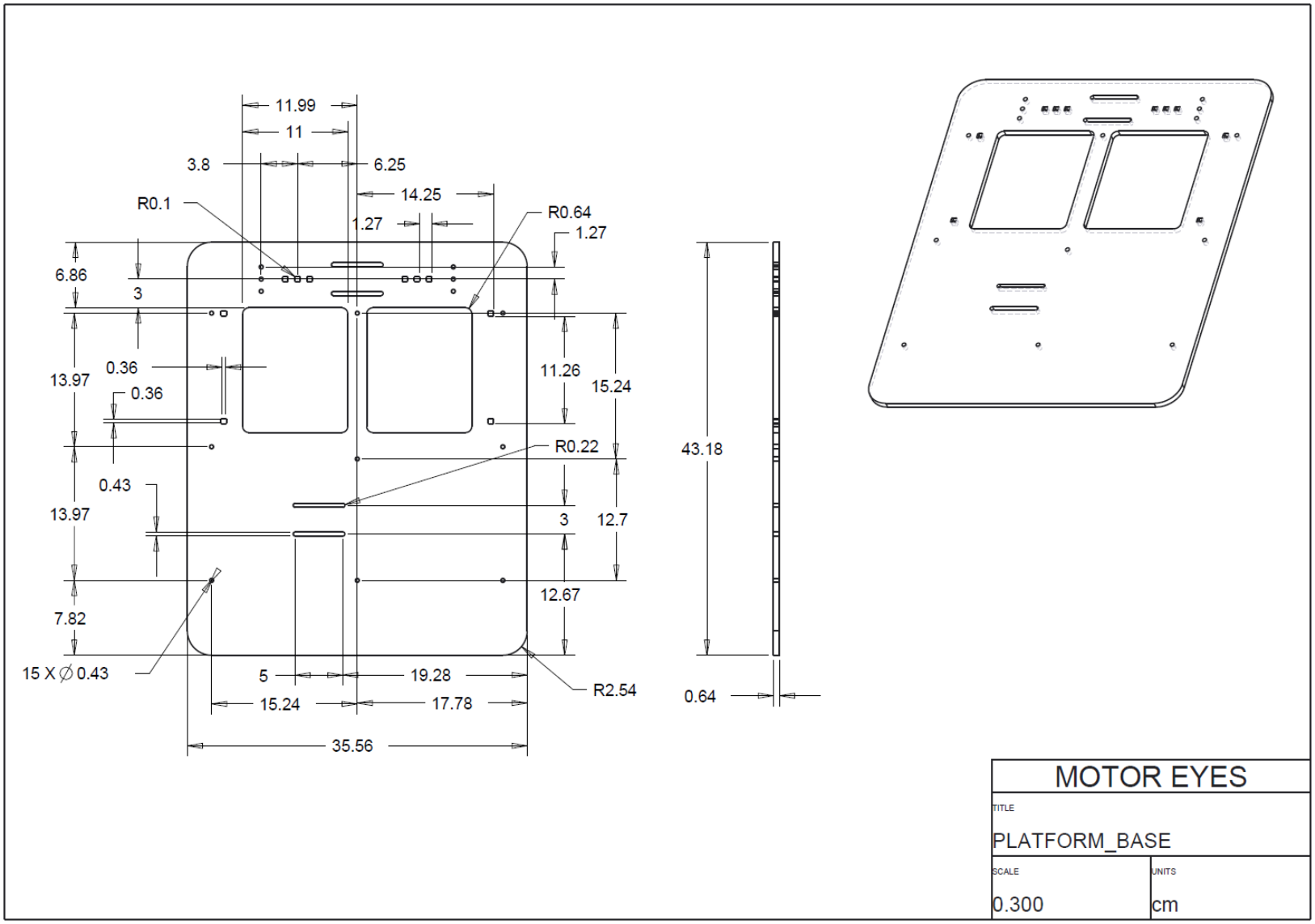
$$fx_{kin} = \frac{(cx \cdot b)}{l_c - b}$$ see Appendix A for derivation

$$fx_{kin} = fz_{actual} \cdot \tan\left(\theta_{panactual} - \theta_{misalign}\right)$$ refer to the above figure

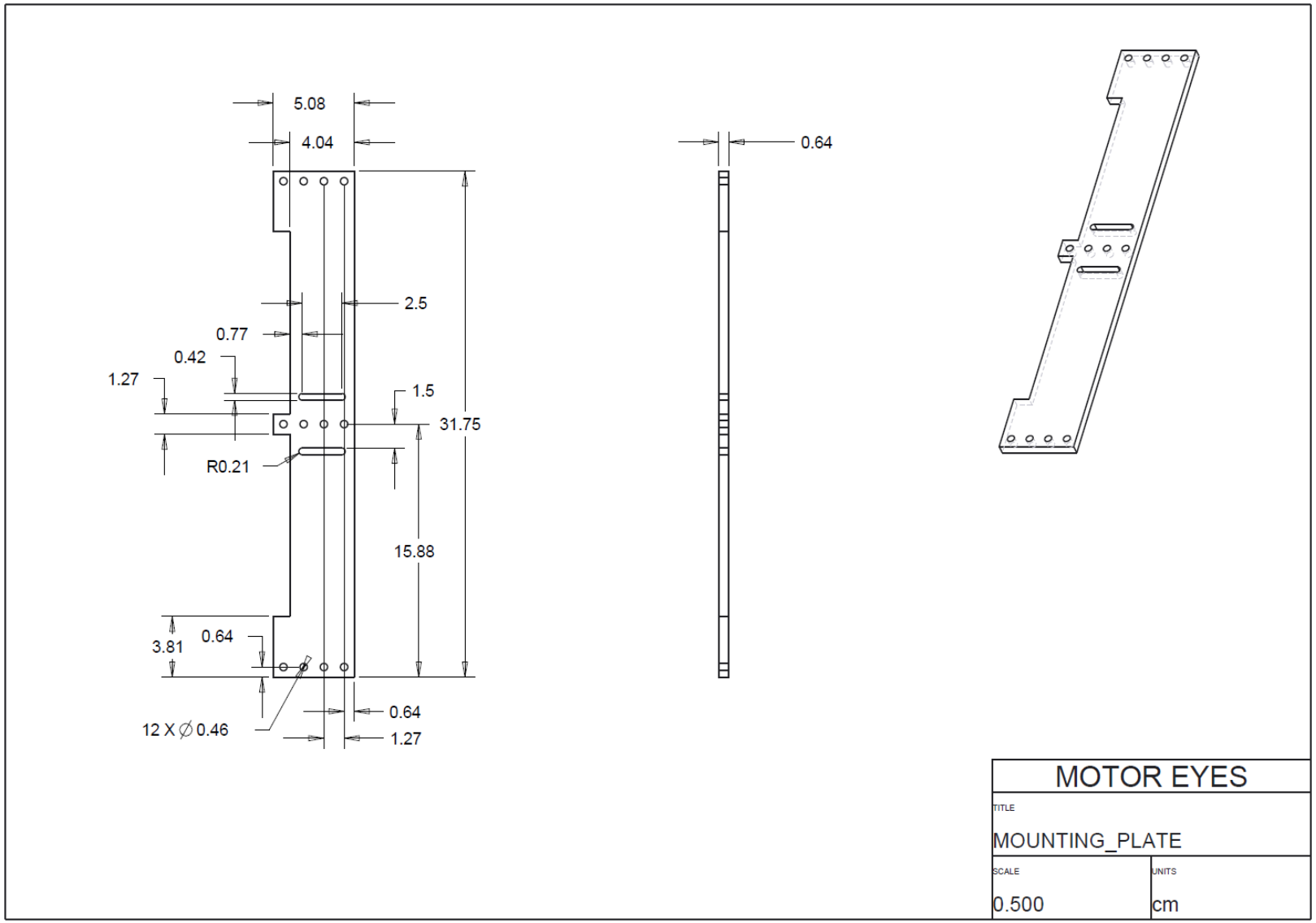
$$\left(\frac{fz_{actual}}{cz_{actual}}\right) = \frac{b}{l_c - b}$$ see Appendix A for derivation

Appendix I: CAD Drawings of Manufactured Parts

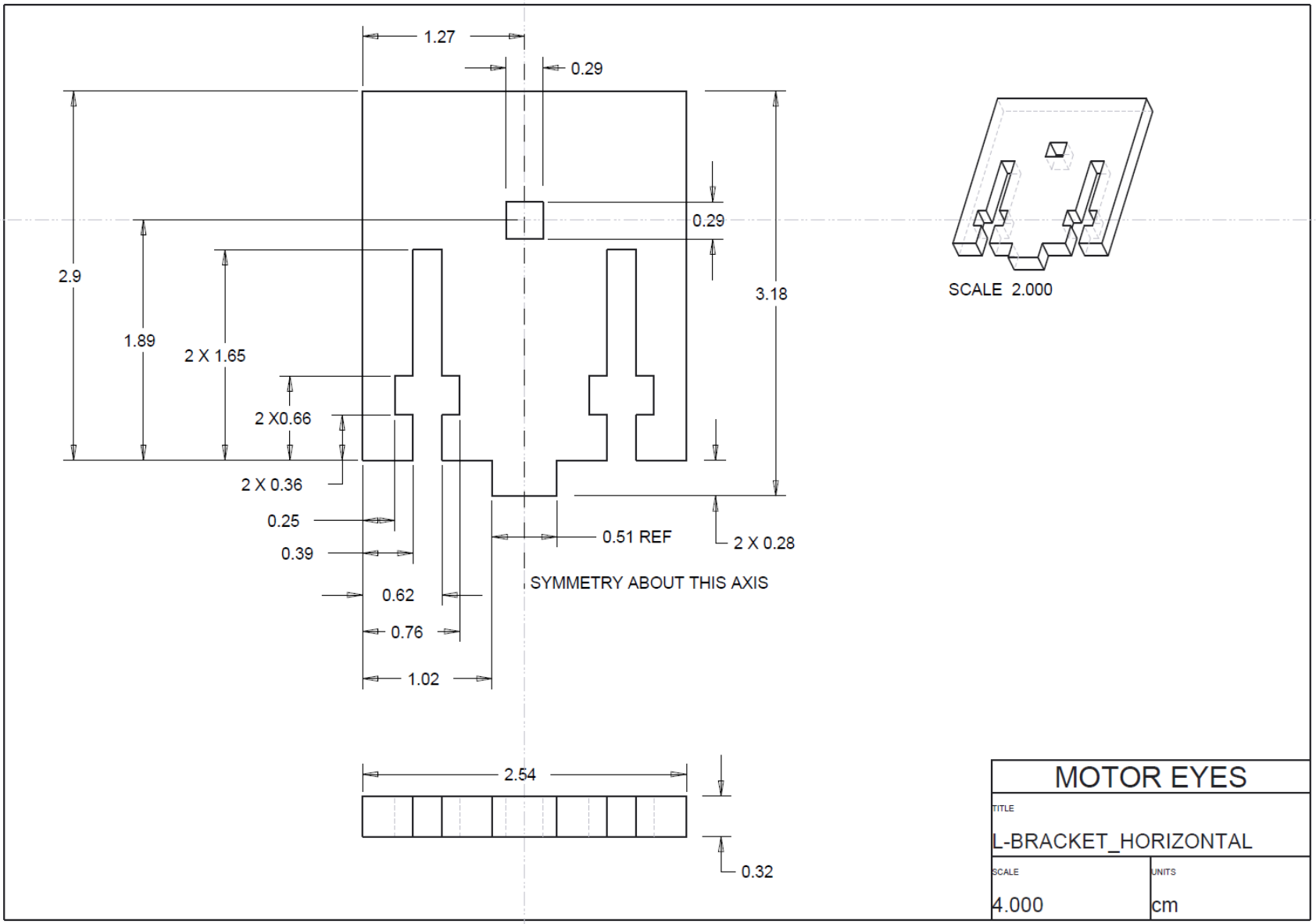
Platform Base



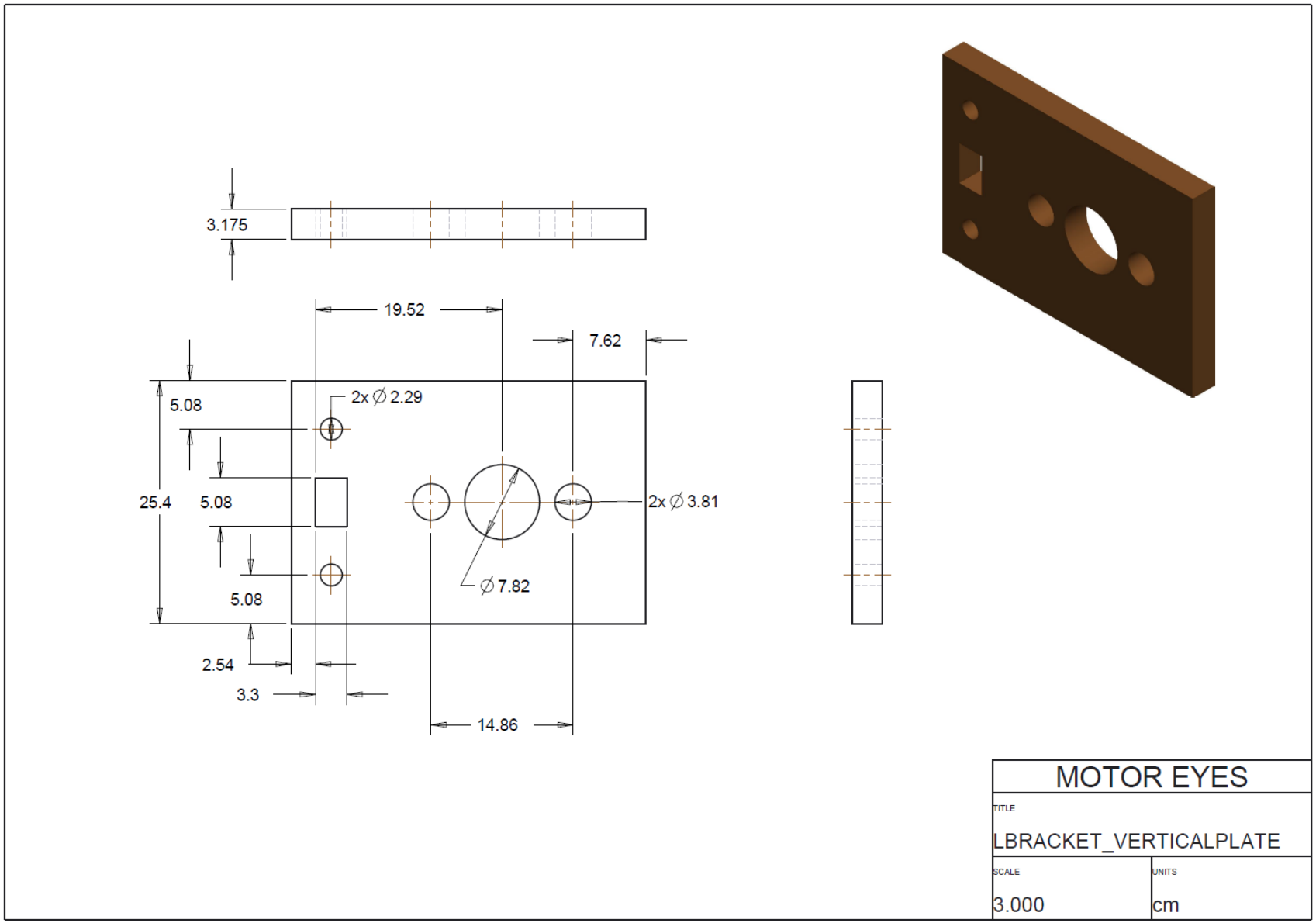
Mounting Plate



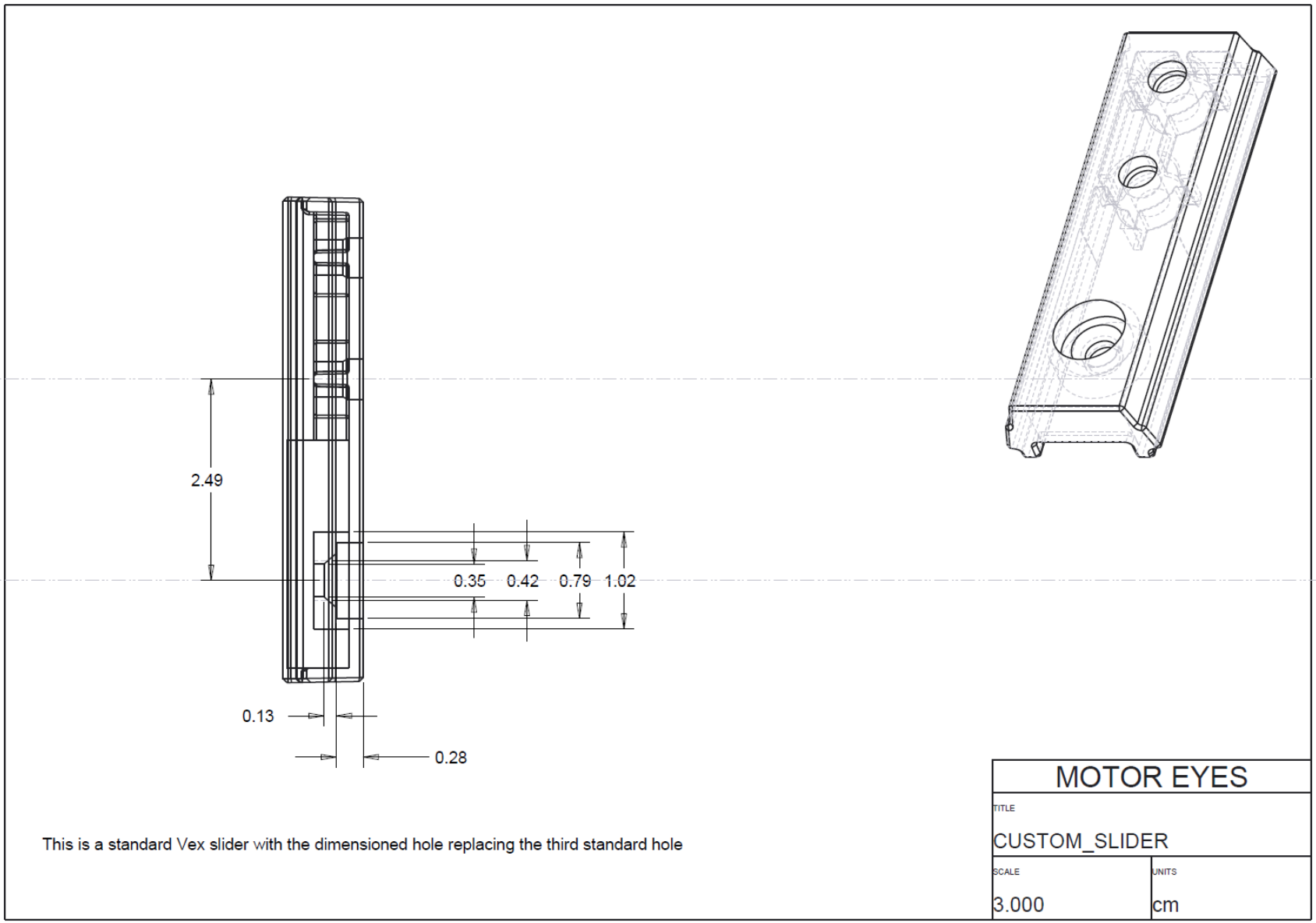
L-Bracket (Horizontal)



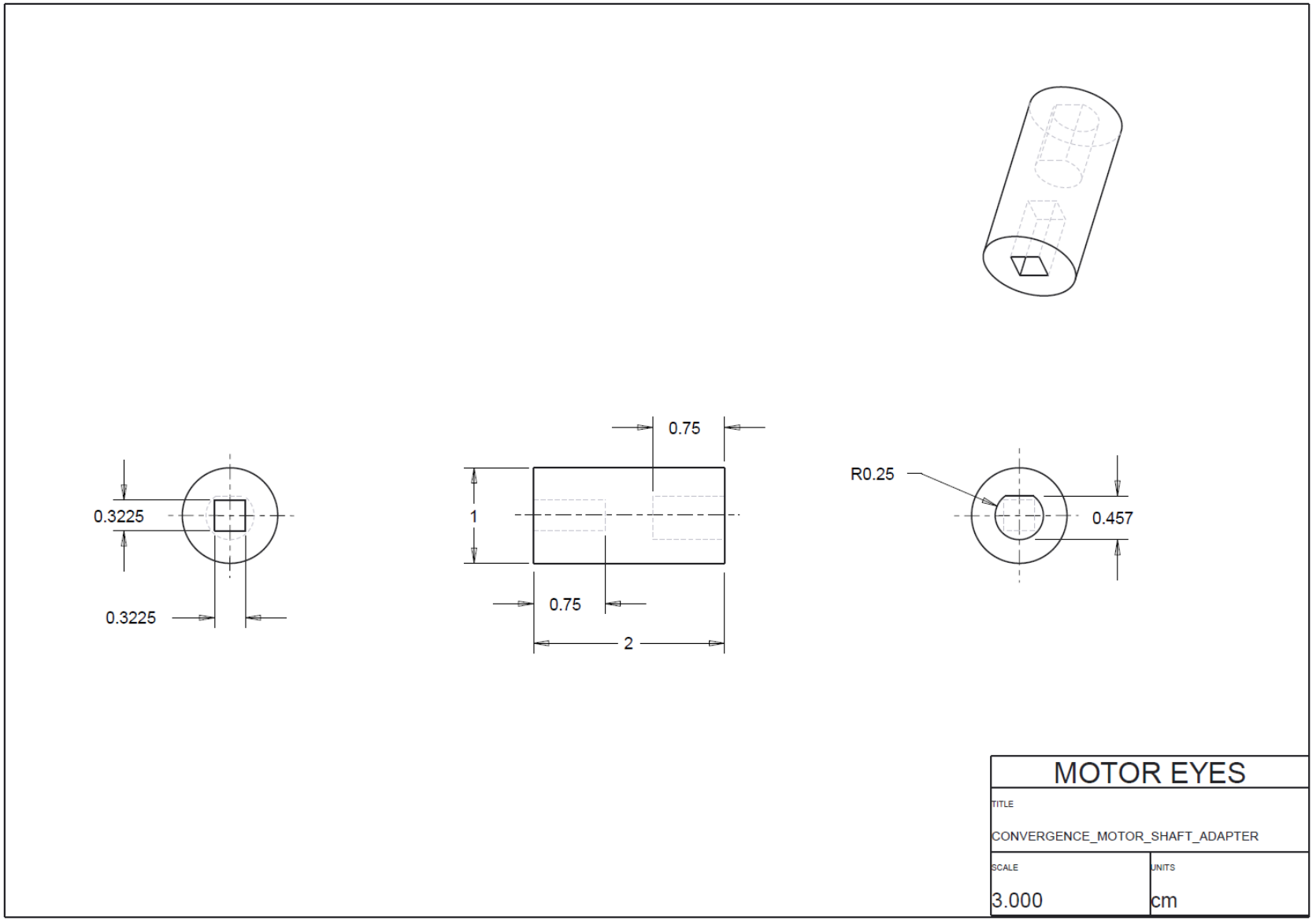
L-Bracket (Vertical)



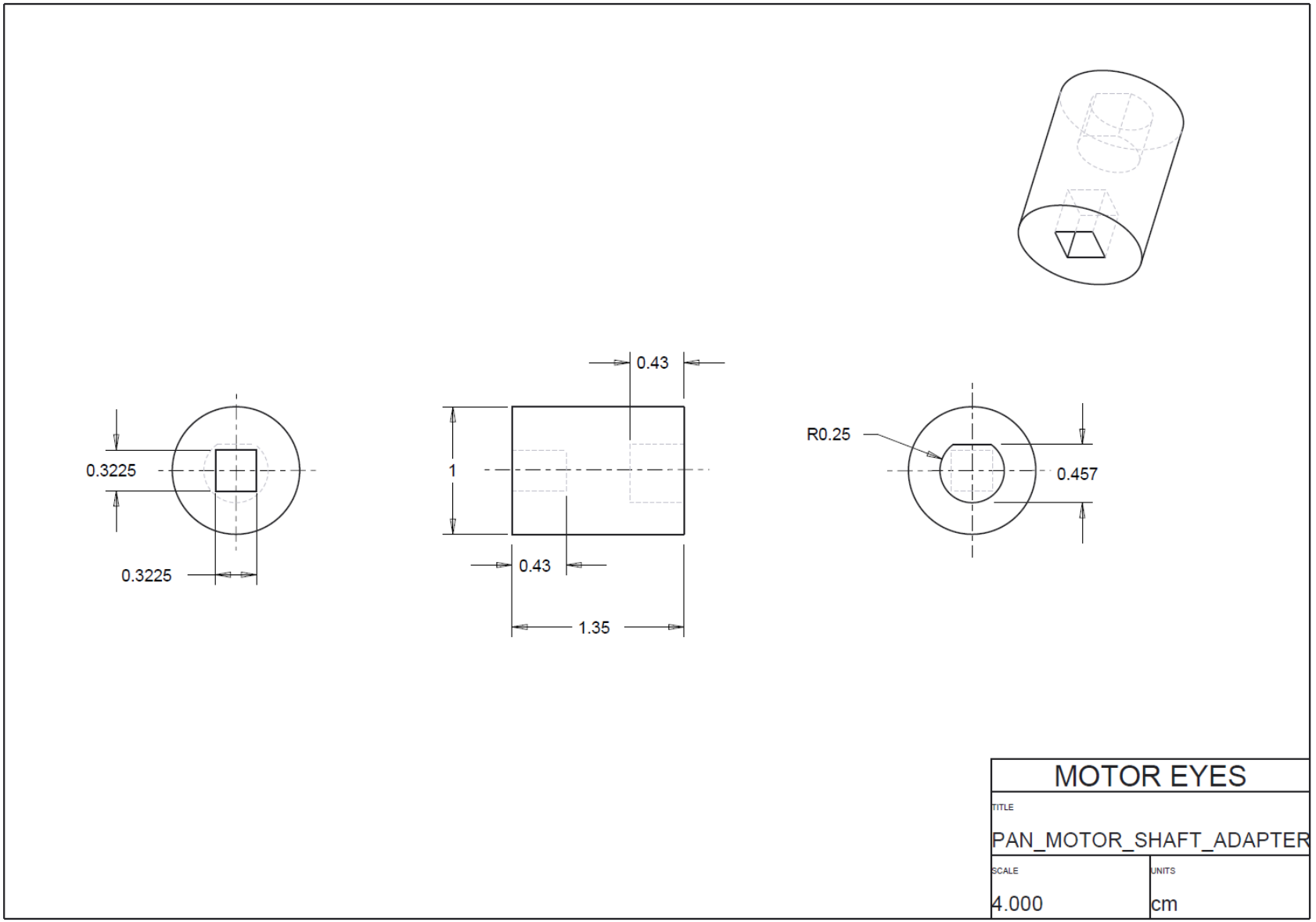
Custom Slider



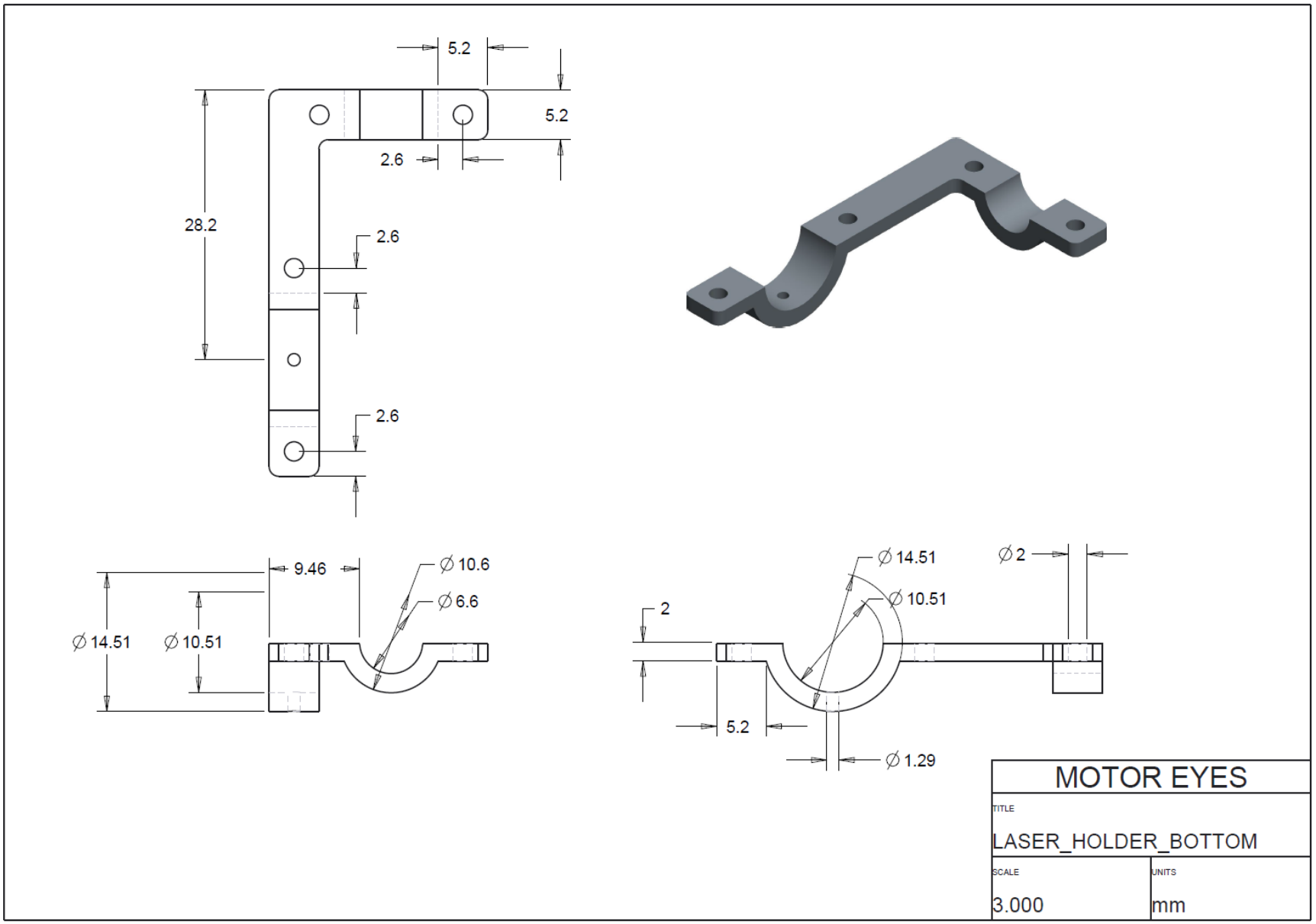
Convergence Motor Shaft Adapter



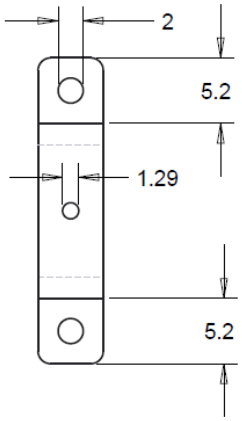
Pan Motor Shaft Adapter

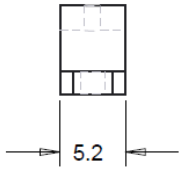


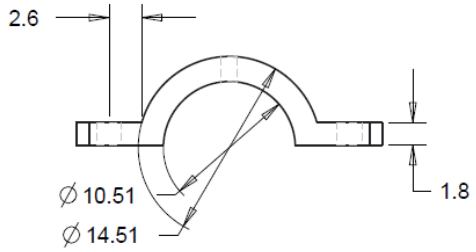
Eye Mount (Part 1: Bottom)



Eye Mount (Part 2: Top – U-Joint)

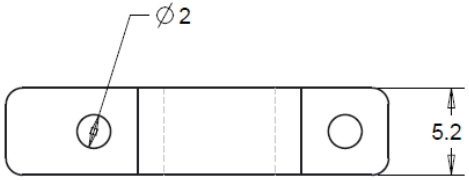


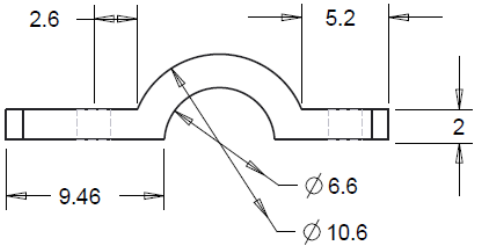




MOTOR EYES	
TITLE	
LASER_HOLDER_TOP_U	
SCALE	UNITS
3.000	mm

Eye Mount (Part 3: Top – Laser)





MOTOR EYES	
TITLE	
LASER_HOLDER_TOP_LASER	
SCALE	UNITS
4.000	cm