# PRECISION POSITION CONTROL USING AN RGB SENSOR AND LINEARIZED OUTPUT VARIABLE-INTENSITY COLOR ARRAY

A Thesis

by

ANDREW CHRISTIAN NELSON

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

| | |
|---|---|
| Chair of Committee, | Won-jong Kim |
| Committee Members, | Bryan P. Rasmussen |
| | Hamid A. Toliyat |
| Head of Department, | Andreas A. Polycarpou |

August 2017

Major Subject: Mechanical Engineering

# ABSTRACT

This thesis presents the design and implementation of a single degree-of-freedom (DOF) position control algorithm, using a red-green-blue (RGB) color sensor and a linearized color array. This work was implemented on a pre-existing magnetic-levitation (maglev) test bed with decoupled controllers for each of the 6 DOFs. The RGB sensor was attached horizontally onto a cantilever beam away from the moving platen and centered over a strip with varying RGB intensity values of the color red. The intensity values that make up the red strip were linearized based upon the response of the sensor as it was moved across the color strip. This linear sensor response was then used to develop a first-order curve-fit to relate position in the *y*-axis to the sensor output. The functionality of the control algorithm was verified through a series of step responses and continuous motion test runs. The position resolution, in the axis of movement, was demonstrated to be better than 30 μm. Experimental results of both the position response and linearization process are presented herein.

# ACKNOWLEDGMENTS

First, I would like to thank my committee chair, Dr. Won-jong Kim, whose advice and guidance have helped me immensely throughout this research. I would also like to thank Dr. Bryan Rasmussen, and Dr. Hamid Toliyat, for serving as my committee members.

I want to extend my gratitude to everyone in the department of mechanical engineering at Texas A&M University, who have taught me everything I know as an engineer, as well as everyone I have had the pleasure of working with. I especially need to thank Dr. Vu Nguyen, whose constant help and guidance along the way have been instrumental in the successful completion of my research.

Thank you to all my friends, and especially my loving girlfriend Amber, who have been by my side during my time at Texas A&M University, who are what made the long days working in the lab bearable.

Finally, I would like to thank my parents, Isabel and Christian Nelson, as well as my siblings Claire, Adam, and James. They have loved and supported me throughout this master's course work.

# CONTRIBUTIONS AND FUNDING SOURCES

This work was supervised by a thesis committee consisting of Professor Won-jong Kim of the Department of mechanical engineering and Professors Bryan Rasmussen and Hamid Toliyat of the Departments of mechanical engineering and electrical engineering, respectively.

The system on which the experiments were completed was initially developed by Dr. Vu Nguyen and Dr. Won-jong Kim. All other work conducted for this thesis was completed by the student independently.

There are no outside funding contributions to acknowledge related to the research and compilation of this document.

# TABLE OF CONTENTS

Page

# LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

## 1.1. Design Objectives

The scope of this thesis is to design, build, and implement a precision position-control system utilizing a red-green-blue (RGB) sensor, and a color strip with varying intensities of a single color. This assembly is then used to provide feedback onto an already existing 6-degree-of-freedom (6-DOF) magnetic-levitation (maglev) platform, to determine its capabilities in providing accurate position feedback and control in one-dimension.

There have been numerous motion-sensing devices used in precision motion control such as rotary encoders and laser interferometers. The downside of these sensors is their level of complexity, as far as system integration and installation, and the fact that they are generally quite expensive. Almost all of these sensors are only accessible through other companies' proprietary services. By using an RGB sensor and linearized color strip, it is possible to achieve 30-micrometer-level resolution with a system that is exceedingly simple to install. This system is a fraction of the expense of the majority of other possible solutions. The system that the analog RGB sensor (TCS3104 manufactured by AMS) was integrated into has a sampling rate of 4 kHz. The rise time of the RGB sensor, which is 12 μs, allows for an accurate sensor response of the current position at each time interval. Overall, the sensor meets all required specifications of the system for precise movement, while minimizing the complexity and cost of the system.

## 1.2. Applications of Linear and Rotary Precision Position Control

High-precision position control has a significant number of applications in industry and is currently an active topic of research. One of the fields most impacted by improvements in the capabilities of high-precision machines is the semiconductor industry. The finer the level of control achieved, by improving the capabilities of the controller, enables more silicon chips to be fabricated at the same time without an increase in the capacity of the machine. The primary use of position control in semiconductor manufacturing is with a process called photolithography. In semiconductors, in order to create complex circuits, a layer of a photo-resistant material is applied to the surface of the silicon substrate in a pattern inverse to the desired pattern on the chip. The wafer is then exposed to ultraviolet (UV) light. This creates a stencil on the wafer, which is then etched using precision tooling. Both the photolithography and the tooling require highly precise motion control in order to repeatedly create working microchips [1]. The primary application of the motion control, in this application, is the step-and-scan movement between individual chip locations on the wafer, which is usually on the scale of 25 nm [2].

Another application of precision motion control is in high-end machining. As the need for machining smaller and more complex parts has increased, the need for precision machining has increased as well. Some computer-numerical-control (CNC) mills use high-precision linear movements to create intricate and complex parts. Conventional CNC milling machines can manufacture with tolerances on the order of 25.4 μm. Improving the capabilities of CNC mills is still an active research area, as there is a high demand for parts on the scale of 100–200 μm [3].

## 1.3. Feedback Mechanisms in Position Control

Precision motion control has been a focus of researchers for many years. Significant work being done in the field has achieved success at controlling motion at the sub-nanometer level by using a variety of methods to control the systems. A meaningful amount of the differentiation among various control systems, outside of their specific control algorithms, is the sensor used to provide positioning feedback. These sensors can be divided into two primary groups: contact and non-contact feedback devices.

The resolution of the feedback sensors is one of the primary limiting factors in the minimum step size a position control system can achieve. Contact based feedback sensors require a physical connection between the moving part and the base on which the global reference frame is located to provide position feedback. An example of this sensor type is linear potentiometers that determine position based on a change in resistance as the slider, in the potentiometer, is moved. Non-contact feedback is the preferred method of position control due to the high resolution and repeatability. Examples of non-contact feedback sensors are laser interferometers and linear variable differential transformers (LVDTs). Four major methods of non-contact feedback control are: (1) capacitive displacement sensors, (2) optical sensor, which are primarily used to detect changes on a surface, (3) detection of magnetic fields using Hall-effect sensors or LVDTs, and (4) laser based feedback.

Capacitive displacement sensors have been used in conjunction with piezo-pneumatic actuators to position with ±0.5 μm accuracy [4]. The sensor works by detecting a change in capacitance across a gap between a sensor and the conductive plate attached to the setup being moved. Since the dielectric constant and the area of the plate remain constant, the change in capacitance is directly proportional to the inverse of the distance between the plate and the

sensor. Optical sensors work by visually detecting changes in a surface, and come in a variety different types and sensing methods. RGB sensors provide feedback based on changes in current due to the optical intensity of the light hitting the sensor head. Optical sensors can also be used to control position from a distance. By shedding a light on a specific point of a system to develop the controller around and placing a focusing lens some distance away, the focused light can be taken into a color sensor and provide positioning data for use in control [5]. Angle-based optical sensors use an optical probe to read the slope of a surface that has a two-dimensional sine wave etched into it as in [6]. By combining an optical sensor and a grid with a repeated sinusoidal pattern across it, an accurate position of the sensor can be determined. Optical sensors are also used in optical scales, which detect slight changes on a finely machined part to achieve resolutions of less than 5 μm [7], with some precision scales being capable of producing nanometer-scale precision. Finally, precision-positioning research with laser interferometers has provided nanoscale precision in positioning. Research applying laser interferometers, as the form of feedback control, has proved quite successful in the past with Nguyen and Kim controlling movement of a maglev system with a resolution of 10 nm [17].

### 1.4. Color Sensing and Motion Control

As the technology of RGB and other similar optical sensors has improved, color sensing for use in various applications has become more popular. The primary application of RGB sensors are target identification and color detection. In the field of object recognition and tracking, it has been shown that color sensors are superior to other current methods, since they are better at locating and tracking an object consistently as it approaches the camera [9]. The drawback of using color sensors, however, is their perceptibility to changes in illumination and orientation. The object-tracking systems are also prone to issues with occlusion or cluttering,

which would be very common in any public setting [9]. Color sensors are also being used in adaptive color-based particle filter tracking. By dividing an image into many fine sections, and applying a color sensor which filters the signal based on specific RGB codes, movement can be tracked over a period of time [10]. Unfortunately, this application has the same drawbacks as the first form of object tracking, mentioned above, in that it is very perceptible to changes in light intensity and the orientation of the target being tracked.

One of the most common applications of color sensors is in color recognition. Color recognition is used in a variety of occupations including in the medical field, manufacturing, and the field of biology. In the medical field, color sensors are being used to detect specific ranges of color values for use in determining blood sugar levels in diabetics. The color sensor was shown to be more effective than the current market tester in detecting changes in blood sugar levels below 100 mg/dL. In some cases, colorimeters using the RGB color sensors were found to have more than a 50% decrease in output error than the commercially available colorimeters [11]. In manufacturing, RGB sensors are being used to identify and sort boxes based on the response of the sensor to a predefined color on the box [12]. Finally, in the field of biology, research is being done using RGB sensors to monitor plant health and growth in a laboratory. A key benefit of using a color sensor for this application is low cost, however, the conclusions drawn from the experiment say that more work will need to be done on optimizing the system before it can be useful [13].

Color- and light-based sensors have also begun to be used in the field of motion control. One such instance is a group that developed a robot that determined its position based upon light-emitting diodes (LEDs) placed on the ceiling. This was done through the use of

triangulation based upon known locations of multiple different colored LEDs. It was found that from a range of 1.95 m, it was possible to locate position down to ±5 cm [14].

## 1.5. Significance of Thesis Contributions

There has been no substantial research done in the field of linear RGB-sensor-based position control. However, work has been done using an RGB sensor to control a rotary motor using the sensor response of a single color as a feedback [15]. Kwon and Kim showed that it is possible to control position of a rotary system using an RGB sensor with a resolution of 0.08°. The research in this thesis was envisioned as a cheaper alternative to more expensive and complex position control methods that have been used in the past to control multiple-DOF position systems. In the case of the testing apparatus that this research was implemented on, laser interferometers and Hall-effect sensors have been used to close the feedback loop. This research investigates the capabilities of the RGB sensor in accurately determining the position of a platen floating over a maglev test bed, using a 1-D array of varying intensities of the color red.

## 1.6. Thesis Overview

This thesis is organized into several chapters discussing the different components of the research. Chapter II discusses the setup of the test bed that is used, including the maglev testing surface, its working principles, and the components involved in the operation of the test bed. Chapter III shows the development of the model of the positioning stage, including the dynamics and force-to-current equations. Chapter IV discusses the development of the color-sensing-based motion control, including, the development of the color strip in both red and blue, the process used to linearize the color strips, and discussion of the limitations and error sources of color-based motion control. Chapter V focuses on the control algorithm and

controller design as well as the process used to integrate the controller into the maglev system. Chapter VI presents the performance of the color-based-control system during a series of tests to assess the capabilities of the sensor. Finally, Chapter VII presents the conclusions drawn from this research as well as possible future work to be done using the RGB sensor.

# CHAPTER II

# MAGLEV TEST BED

## 2.1. Linear Halbach Magnet Array

The test bed on which the RGB-sensor-based position-control algorithm is developed was created by Kim and Nguyen [8][16][17]. It consists of a series of magnets positioned to form a 2-D Halbach array. A simple 1-D Halbach array consists of a series of magnets, with the same length and thickness, laid parallel to each other with the direction of magnetization rotating in one direction by 90° per step. This configuration has been shown by Trumper *et al.* [18] to produce around 90% of the fundamental magnetic field on the strong side of the magnetic array, while the fundamental field on the weak side is virtually eliminated. An example of the magnetic field generated by the magnets in this configuration can be seen in Figure 1.

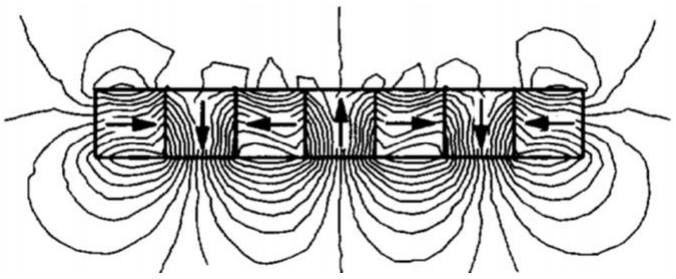


Figure 1: Linear Halbach array with 4 magnet pitch reprinted from [16]

Across the strong side of a Halbach array, the magnetic field produced along the primary axes of the array, in this case the *y* and *z*-axis, is found to have a magnetic flux-density, at a point above the array, equal to

$$B_y(y,z) = \sum_{k=0}^{+\infty} (-1)^k c_n(z) \sin(\gamma_n y) \tag{2.1}$$

$$B_z(y,z) = \sum_{k=0}^{+\infty} (-1)^k c_n(z) \cos(\gamma_n y) \tag{2.2}$$

where $n = 4k+1$ and $\gamma_n = 2\pi n / L$. $c_n(z)$ is a function of the magnetic array and is defined in (3.13) of Chapter III.

In order to create a 2-D magnetic array, with sinusoidally varying magnetic flux densities in both directions, Kim superimposed two linear Halbach arrays perpendicular to one another in the *x*- and *z*-axes [16]. The magnetic matrix consists of cubic magnetic blocks. A 2-D illustration of the strong side of the magnetic array can be seen in Figure 2.

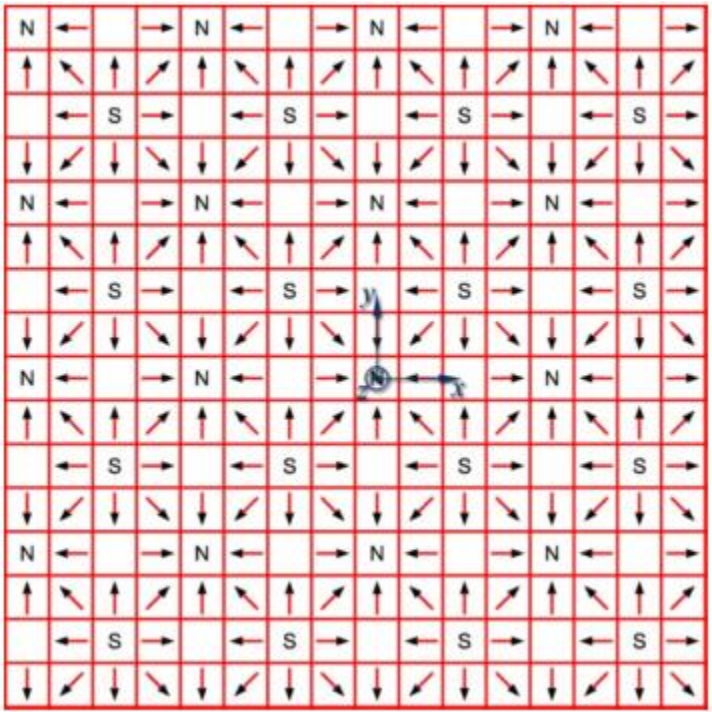



Figure 2: 2-D Halbach array reprinted from [8]

In the two dimensional Halbach array the N and S blocks theoretically have a magnetic remanence $\sqrt{2}$ times stronger than the blocks marked with an arrow. The magnetic field is

directed in the positive *z*-direction for the N blocks, and the negative *z*-direction for the S blocks. The blocks parallel to the N and S blocks have a magnetization vector directed 45˚, with respect to the positive and negative *z*-directions. The diagonal arrows represent blocks with magnetic fields in the various horizontal planes. The strong magnets, ones marked with an N or S, are NdFeB50 with a magnetic remanence of 1.43 T, while the other magnets are NdFeB30 with a remanence of 1.10 T. There are 6 spatial pitches along the *x* and *y* direction with each having a length of L = 0.0508 m [17].

## 2.2. Moving Platen Design

In order to take advantage of the known magnetic flux density a structure (platen) is placed on top of the magnetic array, an image of the platen can be seen in Figure 3.

The platen consists of a square frame with eight coils of wire, distributed with two overlapping coils located at the center of each side of the square. The two coils on each side of the platen have a width of 9.65 mm and an effective length of 42.29 mm. The coils are used to move the platen by passing current through them, the combination of the current through the coils and the magnetic field from the array generates a Lorentz force which is used to move the platen, these resulting force calculations will be discussed in Chapter III.

The coils are positioned in such a way that the long sides of one of the coils, on each side of the platen, is between the long sides of the other coil with a distance of 12.7 mm between each long side of the overlapping coils. The offset, which is equivalent to one quarter of the spatial pitch of the magnetic array, creates a 90° phase difference between each of the long sides of the two coils.

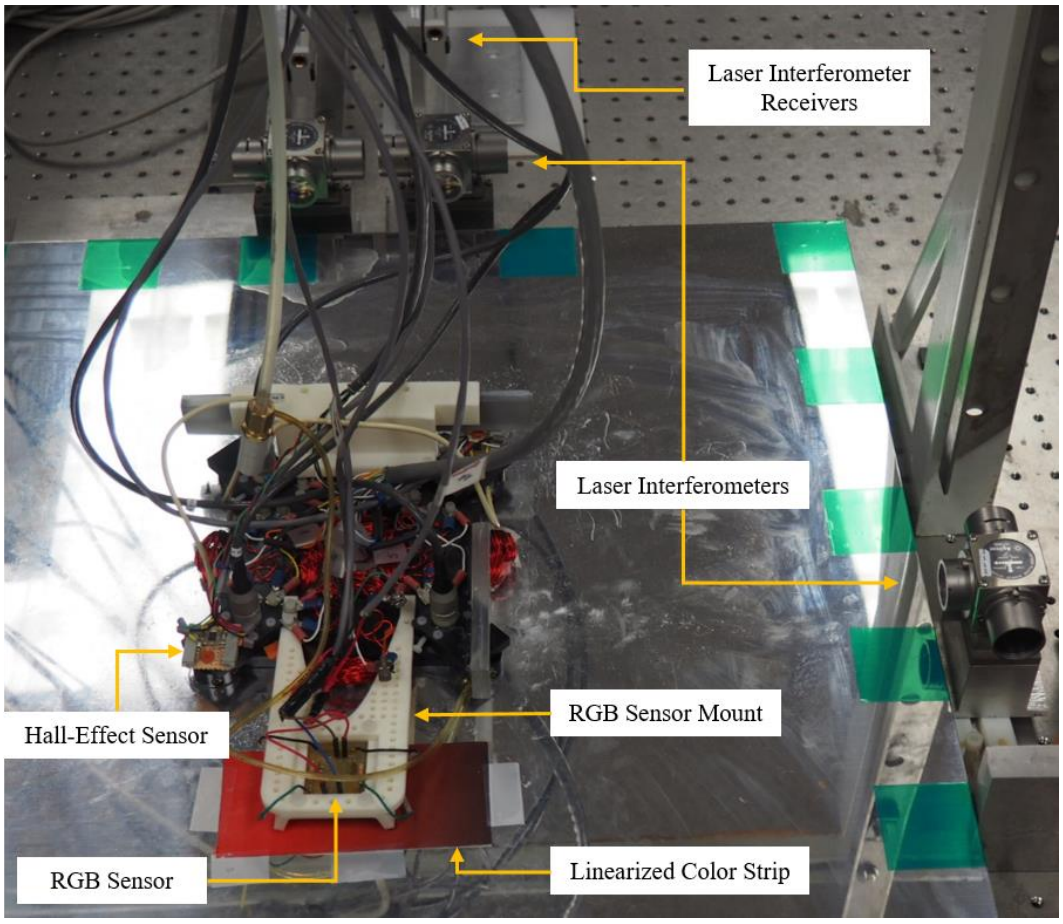



Figure 3: Photograph of experimental setup

This arrangement of coils assures that when one coil produces zero net force, the other coil on the same side of the platen can produce a force with the maximum magnitude possible. If there was only a single coil per side then there would be a point along the sinusoidal magnetic field where the magnetic flux on the long sides of the coil were equal to one another, and thus have a net Lorentz force of zero. Another example is if the coils were to be spaced with an 180˚ phase difference, then there must be a point where the magnetic flux would be equally distributed across both long sides of a single coil. By having the long sides of the coils separated by 90˚ there will always be a coil in a position to generate force. The ends of the coils are also bent up and away from the magnet array in order to limit the end effects [17]. In

order to simplify the control, air bearings (FP-C-010 manufactured by Nelson Air Corporation) are used to provide a constant height in the *z*-direction. The force generated by the coils in creating movement along the *x*- and *y*-axis is not enough in the *z*-direction to affect the height of the platen in any appreciable way.

## 2.3. System Design

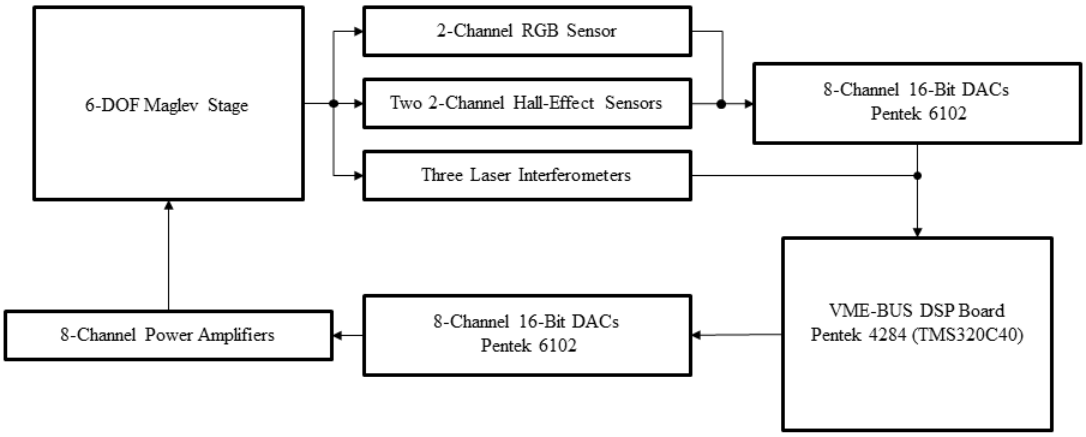



Figure 4: Full system schematic

The schematic of the overall system, used in this research, can be seen in Figure 4. All of the boards used are contained in the Versa-Module-Europa (VME) computer. The VME computer is widely used in control systems for its ability to be easily modified to fit the needs of the user. The VME setup, used for this thesis, includes the Pentek 4284 VMEbus DSP board, which housed the random-access-memory (RAM) and the processor which computes the controls calculations. The VME setup also includes two 8-channel 16-bit digital-to-analog converters (DACs) which are Pentek 6102 I/O boards. The two DACs are used in opposite orientations with one taking the analog input from the sensors and converting them to a digital signal used by the VMEbus, while the other takes the calculated current values sent to the coils and converts the signal from digital to analog. Six channels, of the analog-to-digital converter

(ADC), are used. Four of the ADC channels are used for the two channel Hall-effect sensors, and two are used for the RGB sensor, allowing for simultaneous readings in two different color bands. All eight channels of the DAC board are used, allowing for a current value to be sent to each of the eight separate coils at one time. Due to the setup of the VMEbus interface it is possible to send the position and velocity output of the laser interferometers directly to the DSP board computers, using the prepackaged laser axis boards. The runtime of each experiment is 16.25 s.

During experiments using the RGB sensor as the feedback mechanism, position data is recorded by the redundant laser interferometer. This allows direct comparison between the actual and predicted position of the platen in the *y*-axis.

### 2.4. Feedback Sensors Specifications

There are three types of sensors currently used in the control of the maglev stage. The Hall-effect sensors are used to find the initial position of the platen with respect to the Halbach array. Laser interferometers are used once the initial position has been found to close the feedback loop in the *x* and $\theta_z$ directions. The RGB sensor is used to close the feedback loop in the *y*-direction, and it is the focus of this research. The next few paragraphs will lay out the specifications of each of these sensor types as well as their limitations.

The Hall-effect sensors (2SA-10 manufactured by Sentron) are used to measure the magnetic flux-density of the Halbach matrix. The sensor has an operating voltage of 5 V and can pick up two linear magnetic fields in the *x* and *y*-directions within a range of −40 to 40 mT. The sensitivity of the sensor is 50 V/T. In the test bed, two Hall-effect sensors are placed diagonally on opposite corners of the platen, and offset by either a 90° or 270° phase shift. This

offset is required so that a sensor that is not at a peak of the sinusoidal magnetic field at all times. If there was not an offset between the two sensors, then there would be two positions along the spatial pitch of the magnetic field where the calculated position of the platen would be the same. The phase shift allows for the determination of both the position of the Hall-effect sensor (within one pitch of the magnet array) and the direction that the platen carrying the sensor is moving.

While the RGB sensor is used to close the feedback loop in the *y*-direction of the system, the *x*-direction and moment about $\theta_z$ are closed using feedbacks from two laser interferometers. The two interferometers are placed parallel with the *x*-axis along the top of the test bed, and are separated by a distance of 0.1 m. There is a third laser interferometer placed parallel with the *y*-axis that is used to record the position in the *y*-direction of the platen when moved using the RGB sensor.

The laser head used is the Agilent HP 5517D and the laser interferometers are the Agilent HP 10706B. The laser head emits a HeNe laser beam that is reflected off of aluminum-coated precision mirrors. The mirrors have a surface accuracy of 1/10 of the wavelength of the laser, in this case being approximately 63 nm. The position and velocity output signals, from the laser interferometers, are sent directly to the Laser Axis Board in the VME system.

### 2.5. RGB Sensor Integration

The original maglev system, developed by Nguyen and Kim [8], used only the laser interferometers and Hall-effect sensors to close the feedback loops. In order to integrate the RGB sensor into the system the platen assembly had to be modified. A cantilever mount was developed that could be attached to the platen using pre-existing screw holes in the platen. The

mount was designed in two parts: the boom and the RGB sensor assembly. The boom is a custom part specifically designed for this application and was manufactured from ABS plastic filament using a 3D printer. A model of the boom can be seen in Figure 5. Four holes on the narrow legs of the boom align with threaded holes in the platen to firmly attach the two parts using nylon screws. The boom stretches out past the end of the platen in the *x*-direction, to the area of the surface where the color sensing will take place. It was designed to minimize the additional weight on the platen while still being rigid enough to minimize development of vibrations along the beam as the platen is moved.

The RGB sensor assembly holds the printed-circuit-board (PCB), containing the RGB sensor and 2 light-emitting-diodes (LEDs), as well as the wiring and resistors used to power the RGB sensor and the LEDs. The sensor assembly, which can be seen in Figure 6, was designed to allow for easy access to the PCB and RGB circuit components. The PCB is attached using nylon screws and bolts through four through holes in the part. The center section of the mount was left open to allow for changes to the wiring on the PCB to be done without removal of the part. The size of the mount, with respect to the RGB sensor, is to limit the irradiation of the strip by outside light sources. This was done in an attempt to limit the bias voltage experienced by the sensor due to the ambient light. LED bulbs are fitted into the holes on the angled sides of the mount. The angle of the mount was chosen in order to focus the center of the beam from the LEDs onto the area directly below the RGB sensor, in order to maximize the irradiance of that area.

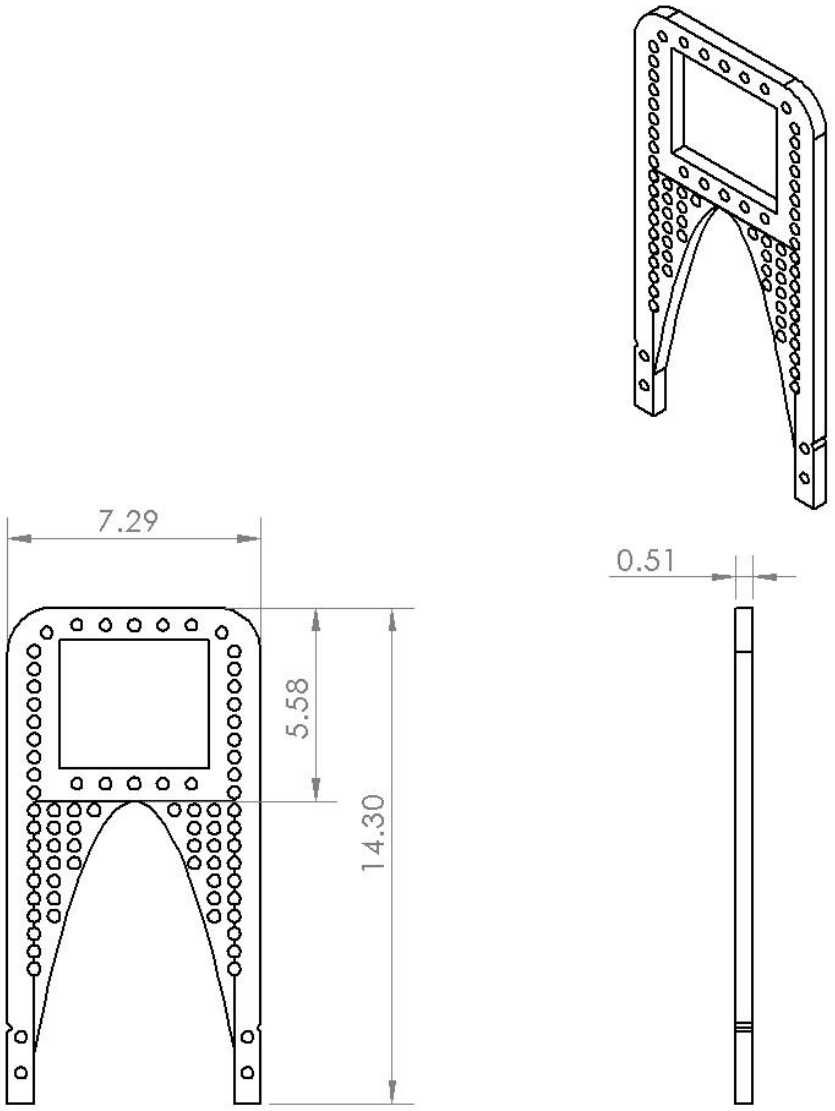



Figure 5: Boom component of mount assembly with units in cm

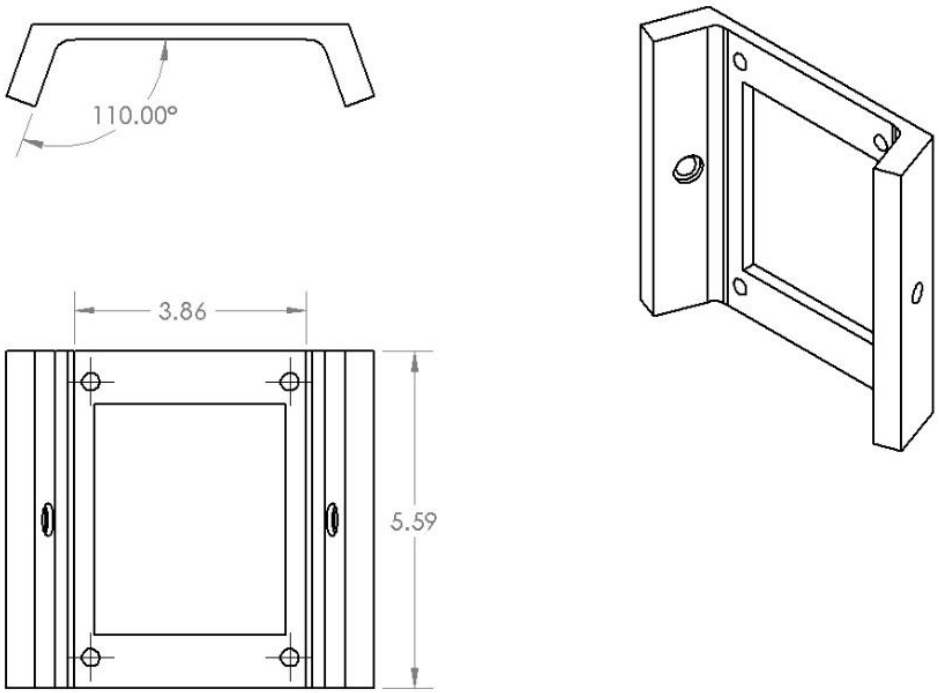



Figure 6: RGB sensor mount

The RGB sensor used in this research is the TCS3104 manufactured by AMS. It is a three-channel RGB color sensor that converts light intensity to voltage. The operating voltage is 5 V and it has an irradiance responsivity ($r_e$) in the red color frame of 0.02–0.4 mV/μW/cm$^2$ based on the wavelength of color. The rise time of the sensor is approximately 12 μs. The sensor and its inputs and outputs can be seen in Figure 7. The multi-function logic input is used to add either a gain factor of 1× when not engaged, or a gain output of 4× when voltage is supplied, in order to modify the range of the output voltage. A 0.1-μF decoupling capacitor was placed between the power and ground lines of the RGB sensor in order limit noise in the sensor output from changes in the current drawn from the batteries. The RGB sensor is approximately 2 mm square with a height of 0.65 mm.

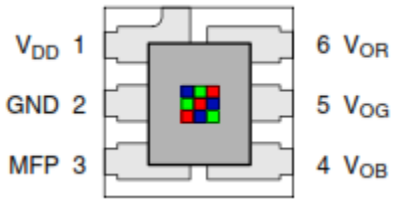



Figure 7: RGB sensor

The LEDs are the YSL-R547W2C-A13 manufactured by China Young Sun LED Technology. Each LED has a maximum forward current of 20 mA and a luminous intensity of 90000 mcd. The viewing angle at half power is ±10°. In order to maximize the light, on the specific area of interest of the color strip, the LEDs were attached at 8° angles to the mount, and directed towards the area of interest on the color strip. 330-Ω resistors were used to create a current of 13.6 mA to the LED, as that was found to be the level of brightness delivering the most consistent results. Both the LEDs and the RGB sensor were powered using four AA batteries providing 4.5 Vdc, which is within the desired working range of both components.

To improve the output from the Hall-effect and RGB sensors, an analog first order RC low-pass filter was added between the signal output and the ADC converter in the VME chassis. The corner frequency of the RC filter is 137.2 Hz, and the transfer function can be seen in (2.3).

$$L_y(s) = \frac{1}{0.00116s + 1} \tag{2.3}$$

A digital low-pass filter was also implemented to help eliminate noise from the output of the RGB sensor. The filters were designed based upon the closed loop bandwidth of the system of 14 Hz. The digital filter's continuous time transfer function is

$$L_y(s) = \frac{1}{0.002s + 1}. \tag{2.4}$$

This equates to a corner frequency of 79.58 Hz.

Prior to the implementation of the two first-order low-pass filters the noise in the signal was large enough to cause the system to become unstable when the RGB sensor was used to close the feedback loop. With the final controller iteration, it was found that the average peak-to-peak noise amplitude in the signal is approximately 200 μm. This noise is larger than desired, but in order to get relatively fast and accurate movements, the gain in the system had to be increased leading to an increased noise amplitude in the signal. In order to show that the noise originates from the RGB sensor and not the dynamics of the system, the system was held at the origin throughout the run where the noise produced would be solely a function of the RGB sensor output. The plot of this steady-state noise can be seen in Figure 8. Comparisons between controllers with different design specifications in terms of noise versus rise time will be shown in Chapter VI, where the difference in noise amplitude between the two will be shown.

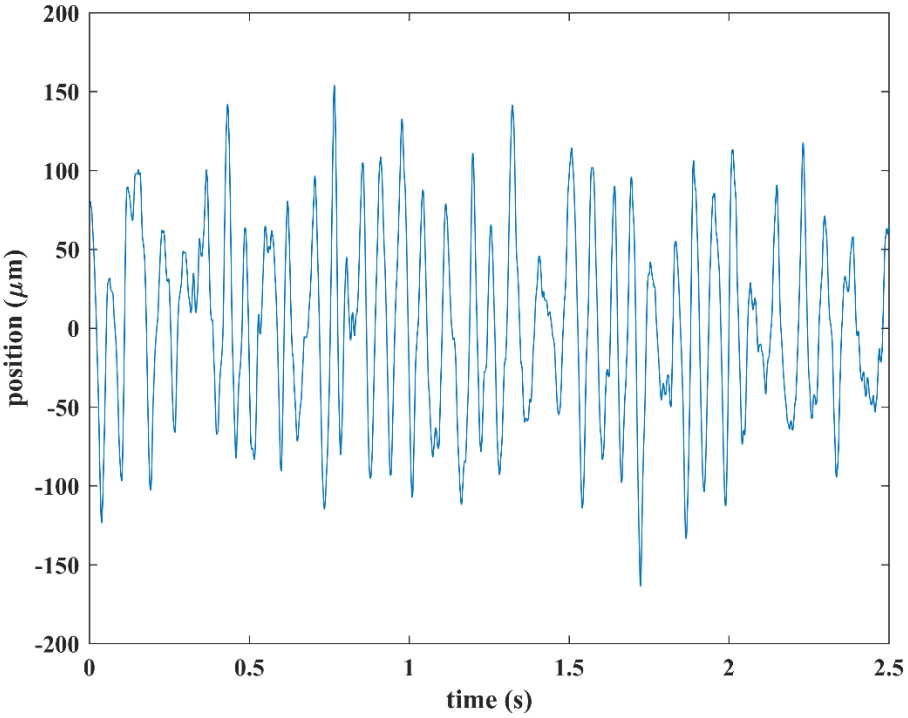



Figure 8: Noise amplitude of RGB sensor

# CHAPTER III

# MODELING OF POSITIONING STAGE

## 3.1. System Dynamics and Modeling

The only moving part of the maglev stage, is the platen which can be modeled as a pure mass with force being applied from the Lorentz force generated by the coils and underlying magnetic matrix. An engineering drawing of the platen can be seen in Figure 9.

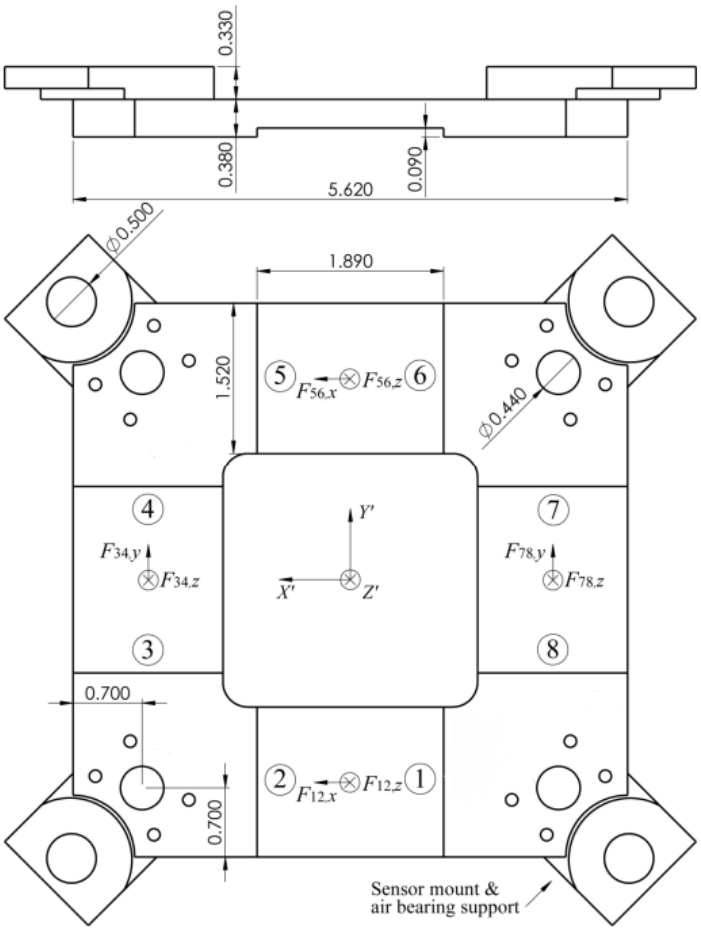



Figure 9: Engineering drawing of moving platen with units in cm adapted from [17]

With the arrangement of forcers shown in the engineering drawing above, each primary axis of motion can be controlled using 2 sets of forcers, with the exception of the *z*-axis. Forcers 1, 2 and 5, 6 generate force in the *x*-axis and can be used to rotate about *X′*. Forcers 3, 4 and 7, 8 generate force in the *y*-axis and can be used to rotate about *Y′*. To generate a force in the *z*-axis, or rotation about *Z′*, components of the force from all of the forcers would need to be used. The direction of the current through each coil, when viewed from above is as follows: coils 1, 3, 5, and 7 have current flowing in the clockwise directions, while coils 2, 4, 6, and 8 have current flowing in the counterclockwise direction.

The Newtonian dynamics of the system are as follows:

$$m\ddot{x} = F_x \tag{3.1}$$

$$m\ddot{y} = F_y \tag{3.2}$$

$$m\ddot{z} = F_z \tag{3.3}$$

$$I_x\ddot{\theta}_x = T_x \tag{3.4}$$

$$I_y\ddot{\theta}_y = T_y \tag{3.5}$$

$$I_z\ddot{\theta}_z = T_z \tag{3.6}$$

In the above equations, *x*, *y*, and *z* are the position in the *X*, *Y*, and *Z*-axes, which is the coordinate frame about the magnetic test bed. $\theta_x, \theta_y$ and $\theta_z$ represent the angle which the platen is rotated about *X′*, *Y′*, and *Z′*, which is the coordinate frame with respect to the platen. The mass of the system, including the mirrors, RGB mount, wires, and other components was

found to be 0.90 kg. The estimated moment of inertia with respect to the *X′*, *Y′*, and *Z′*-axes is $I_x = 0.001975\,\text{kg-m}^2$, $I_y = 0.003026\,\text{kg-m}^2$, and $I_z = 0.001205\,\text{kg-m}^2$, respectively. The resultant forces generated due to the Lorentz force between the current through the coils and the magnetic test bed, acting in the three primary axes are $F_x$, $F_y$, and $F_z$. The resultant torques generated by the Lorentz force in about the three axes are $T_x$, $T_y$, and $T_z$.

### 3.2. System Dynamics and Force Generation

In order to develop a set of equations to model the system, a set of relationships between the Lorentz force generated in the individual coils as they relate to the force generated in the 6-axis of motion must be developed. The steps to show these relationships will now be discussed. The force generated by the coils in the *x* and y-axis are the same, and only the equation for the force in the *y*-axis will be shown. Also, even though, for this thesis, the z-axis was maintained at a constant value, using air bearings, in order to fully describe the system the equations describing the force generated in the *z*-axis will still be included. The relationship between the coil pairs discussed above and the force generation in the primary axis of motion are

$$F_{1,y} = -J_1 pc_1(z)\frac{4}{\gamma_1^2}\cos\left[\gamma_1\left(y-\frac{a}{2}\right)\right]\sin\left(\frac{\gamma_1 a}{2}\right)(1-e^{-\gamma_1 d}) \tag{3.7}$$

$$F_{1,z} = J_1 pc_1(z)\frac{4}{\gamma_1^2}\sin\left[\gamma_1\left(y-\frac{a}{2}\right)\right]\sin\left(\frac{\gamma_1 a}{2}\right)(1-e^{-\gamma_1 d}) \tag{3.8}$$

$$F_{2,y} = -J_2 pc_1(z)\frac{4}{\gamma_1^2}\sin\left[\gamma_1\left(y-\frac{a}{2}\right)\right]\sin\left(\frac{\gamma_1 a}{2}\right)(1-e^{-\gamma_1 d}) \tag{3.9}$$

$$F_{2,z} = J_2 p c_1(z) \frac{4}{\gamma_1^2} \cos\left[\gamma_1\left(y - \frac{a}{2}\right)\right] \sin\left(\frac{\gamma_1 a}{2}\right)(1 - e^{-\gamma_1 d}) \tag{3.10}$$

$$F_{12,y} = b_1(z)\{-J_1 \cos\left[\gamma_1\left(y - \frac{a}{2}\right)\right] + J_2 \sin\left[\gamma_1\left(y - \frac{a}{2}\right)\right]\} \tag{3.11}$$

$$F_{12,z} = b_1(z)\{J_1 \sin\left[\gamma_1\left(y - \frac{a}{2}\right)\right] + J_2 \cos\left[\gamma_1\left(y - \frac{a}{2}\right)\right]\} \tag{3.12}$$

The derivations of these force equations can be found in the Nguyen's dissertation [17], in which the maglev system was originally developed. The force, generated by the interactions between the magnetic array and the current through the coils, is a function of the coil properties and the magnetic field. Each coil side has $N = 30$ turns stacked in three layers. The thickness, width, and effective length of each coil side is $d = 2.54\,\text{mm}$, $a = 9.65\,\text{mm}$, and $p = 42.29\,\text{mm}$ respectively. $J_n$ (A/m$^2$) is the current density along the primary axis of the coil. The special pitch of the sine wave, in the *yz* and *xz*-axis, of the magnet arrays is $L = 0.0508\,\text{m}, n = 4k + 1$ and $\gamma_n = 2\pi n / L$. $c_n(z)$ and $b_1(z)$ are both considered constant due to the relatively small constant height in the *z*-axis, $c_n(z)$ is a function of the magnetic array, while $b_1(z)$ relates the magnetic array with the current through the coils. Both functions are calculated as follows [17]

$$c_n(z) = \frac{2\sqrt{2}\mu_0 M_0}{\pi n}(1 - e^{-\gamma_n \Delta}) e^{-\gamma_n z} \tag{3.13}$$

$$b_1(z) = p c_1(z) \frac{4}{\gamma_1^2} \sin\left(\frac{\gamma_1 a}{2}\right)(1 - e^{-\gamma_1 d}) \tag{3.14}$$

$\mu_0 = 4\pi \times 10^{-7}$ H and is the permeability of free space, $M_0 = 1.43$ T is the peak magnetization of the magnets, and $\Delta = 12.7$ mm is the thickness of the magnet array along the *z*-axis.

Next, the relationship between the forces generated in the individual coil sets and the current required to generate them needs to be determined. As discussed earlier, the force generated in each of the primary axis is made up of the force generated in multiple coil sets. These relationships are [17]

$$\begin{bmatrix} F_x \\ F_y \\ T_z \\ F_z \\ T_x \\ T_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ l & 0 & l & 0 & -l & 0 & -l & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & -l & 0 & 0 & 0 & l & 0 & 0 \\ 0 & 0 & 0 & -l & 0 & 0 & 0 & l \end{bmatrix} \begin{bmatrix} F_{12,x} \\ F_{12,z} \\ F_{34,y} \\ F_{34,z} \\ F_{56,x} \\ F_{56,z} \\ F_{78,y} \\ F_{78,z} \end{bmatrix} = A \begin{bmatrix} F_{12,x} \\ F_{12,z} \\ F_{34,y} \\ F_{34,z} \\ F_{56,x} \\ F_{56,z} \\ F_{78,y} \\ F_{78,z} \end{bmatrix} \tag{3.15}$$

where $l = 0.05207$ m, which is the distance between the midpoints of each coil side with respect to the platen's axis perpendicular to said coil side.

The force-current transformation, between the current through each coil and the force generated, was found to be [17]

$$\begin{bmatrix} F_{12,x} \\ F_{12,z} \\ F_{34,y} \\ F_{34,z} \\ F_{56,x} \\ F_{56,z} \\ F_{78,y} \\ F_{78,z} \end{bmatrix} = b_*(z) \begin{bmatrix} sx & -cx & 0 & 0 & 0 & 0 & 0 & 0 \\ cx & sx & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & cy & sy & 0 & 0 & 0 & 0 \\ 0 & 0 & -sy & cy & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & cx & -sx & 0 & 0 \\ 0 & 0 & 0 & 0 & -sx & -cx & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -sy & -cy \\ 0 & 0 & 0 & 0 & 0 & 0 & -cy & sy \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ i_7 \\ i_8 \end{bmatrix} \tag{3.16}$$

where, $sx = \sin(x_2)$, $cx = \cos(x_2)$, $sy = \sin(y_8)$, and $cy = \sin(y_8)$. Also, $x_2$ and $y_8$ are the base points along the *X*- and *Y*-axis about which the force calculations are done. These base points are located on coils 2 and 8. The exact location of these two points is inconsequential as long as they are located at the same relative position on each coil. The variable $b_*(z_0)$ is a function of the coils and magnetic field and is considered to be a constant value of 0.85 throughout the run.

The control effort for each degree of freedom is related to the individual force components as shown in (3.17).

$$\begin{bmatrix} F_{12,x} \\ F_{12,z} \\ F_{34,y} \\ F_{34,z} \\ F_{56,x} \\ F_{56,z} \\ F_{78,y} \\ F_{78,z} \end{bmatrix} = 2Bb_*(z) \begin{bmatrix} u_x \\ u_y \\ u_{\theta z} \\ u_z \\ u_{\theta x} \\ u_{\theta y} \end{bmatrix} \tag{3.17}$$

Here, *B* is the pseudo-inverse of matrix *A* and is calculated to be [17]

$$B = \begin{bmatrix} 0.5 & 0 & 4.8012 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & -9.6025 & 0 \\ 0 & 0.5 & 4.8012 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & -9.6025 \\ 0.5 & 0 & -4.8012 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 9.6025 & 0 \\ 0 & 0.5 & -4.8012 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 9.6025 \end{bmatrix} \tag{3.18}$$

Through manipulation of the above equations the relationship between current and control input can be determined as shown in (3.19).

$$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ i_7 \\ i_8 \end{bmatrix} = 2B\frac{1}{b_*(z_0)}\begin{bmatrix} sx & cx & 0 & 0 & 0 & 0 & 0 & 0 \\ -cx & sx & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & cy & -sy & 0 & 0 & 0 & 0 \\ 0 & 0 & sy & cy & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & cx & -sx & 0 & 0 \\ 0 & 0 & 0 & 0 & -sx & -cx & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -sy & -cy \\ 0 & 0 & 0 & 0 & 0 & 0 & -cy & sy \end{bmatrix}\begin{bmatrix} u_x \\ u_y \\ u_{\theta z} \\ u_z \\ u_{\theta x} \\ u_{\theta y} \end{bmatrix} \tag{3.19}$$

The values of the sine and cosine components can be determined in real-time using the position output from the laser interferometers and RGB sensors to calculate the values of $x_2$ and $y_8$. From the above derivation, it can be seen that each of the 6 DOFs can be decoupled from the rest and controlled individually. This allows for the sensor, used for position control in each axis, to be changed and modified independent from the rest of the system.

# CHAPTER IV

# COLOR-SENSING-BASED MOTION CONTROL

This chapter fully describes the color-sensing components of the overall system. Specifically, the dynamic model of the color-sensing circuit, the approximate radiant power transferred between the LEDs and the RGB sensor, and the development of the linearized color strip in both red and blue are presented.

## 4.1. Modeling of RGB sensor and LEDs

In order to insure there is no unintentional loss of data between the RGB sensor and ADCs, it is necessary to model the dynamics of the LEDs and RGB sensor. The RGB sensor can be modeled using a transfer function relating the output voltage, $V_o$, of the RGB sensor to the reverse-bias current, $I_p$, caused by the light intensity from the LEDs, as [15][19]

$$V_o(s) = \frac{R_r}{\tau_{RC} + 1} I_p(s) \tag{4.1}$$

where $R_r$ is the resistance in the RGB amplifier circuit and $\tau_{RC}$ is the time constant of the amplifier circuit. Using the known rise time of the RGB sensor of 12 μs the time constant of the RGB amplifier circuit is found to be $\tau_{RC}$ = 5.5 μs.

The other variable to take into account, when examining the circuit, is the propagation delay time or the time it takes the light to travel to the sensor. The propagation delay time can be calculated as [15]

$$\tau_d = \frac{d}{c} \tag{4.2}$$

where $c$ is the speed of light and $d = d_1 + d_2$, from Figure 10 below. The propagation delay time constant for this setup was found to be $\tau_d = 7.84x10^{-11}$ s.

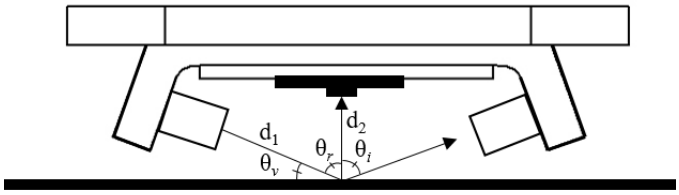



Figure 10: Relationship between LEDs and RGB sensor

With the time constants from (4.1)–(4.2) the bandwidth of the light propagation mechanism can be approximated. The bandwidth is a function of the sum of the two time constants. However, the bandwidth is dominated by the RGB amplifier circuit, due to how small the propagation delay time constant is in comparison. The bandwidth is calculated as [15]

$$f_{BW} \approx \frac{1}{2\pi(\tau_d + \tau_{RC})} \tag{4.3}$$

From (4.3) the predicted bandwidth is found to be 28.33 kHz. The large bandwidth of the RGB circuit, when compared to the low bandwidth of the closed-loop controller, which will be discussed later, allows for a couple of simplification to be made in the system model. First, the propagation delay can be considered negligible when determining the sensor output. Second, the fast response of the sensor allows for it to be modeled as a constant value used to relate the voltage output to the current generated by the photodiode as [15]

$$v_o = R_r i_p \tag{4.4}$$

It is possible to make these simplifications because any changes in the RGB sensor will occur much quicker than in the controller, which samples at 4 kHz.

## 4.2. Radiant Power Model

Along with the modeling of the RGB and LED sensors, in order to fully describe the system, the radiant power being transferred from the LEDs to the sensor face must be quantified. It is assumed that the LED has a Lambertian radiation pattern, meaning the radiant intensity observed from an ideal reflecting surface is directly proportional to the cosine of the angle between the direction of the incident light and the surface normal [20].

The radiant intensity, with respect to the viewing angle $\theta_v$ shown in Figure 10, is calculated as [15]

$$I_e(\theta_v) = P_t\left[\frac{m+1}{2\pi}\cos^m\left(\theta_v\right)\right] \tag{4.5}$$

Where $P_t$ is the optical power from the LED and the semi-angle at half power is $m$ = ln 2/ln(cos($\Phi_{1/2}$)) [21], $\Phi_{1/2}$ is the half power viewing angle and is a property of the LED bulb.

To determine the received irradiance to the sensor, it is first necessary to quantify the irradiation on the strip, and subsequently the reflected irradiation that hits the sensor face. The irradiance and received optical power from the LED to the strip can be shown to be [15]

$$E_{ts}(\theta_v) = P_t\left[\frac{m+1}{2\pi}\cos^m\left(\theta_v\right)\right]\frac{\cos\left(\theta_i\right)}{d_1^2} \tag{4.6}$$

$$dP_{ts}\left(\theta_v\right) = E_{ts}\left(\theta_v\right)dA_I \tag{4.7}$$

Where $\theta_i$ is the incident angle of the light and $dA_I$ is the incident-beam area modeled as a point.

In order to determine the received irradiation at the sensor face the power emitted off of the surface, found in (4.7), is now considered as the light source. The emitted power from the color strip is a function of the Lambertian irradiance, with the addition of specular irradiance caused by imperfections in the reflective surface of the color strip. The power emitted from the color strip and the specular irradiance can be calculated as [15]:

$$dP_{rs}\left(\theta_v,\theta_r,\theta_a\right)=\rho\left(E_{ts}\left(\theta_v\right)+E_{ss}(\theta_v,\theta_r,\theta_a)\right)dA_i \tag{4.8}$$

$$E_{ss}(\theta_i,\theta_r,\theta_a)=\alpha\exp\left[-\frac{1}{2}\left(\left(\frac{\theta_i-\theta_r}{\sigma_s}\right)+\left(\frac{\theta_a}{2\sigma_s}\right)^2\right)\right] \tag{4.9}$$

$$\alpha=\frac{I_e k_s}{\left(d_1^2\sqrt{2\pi\sigma_s}\right)} \tag{4.10}$$

Here, $\rho$ is the reflectance factor, $\theta_r$ is the angle between the LED and sensor face, and $\theta_a$ is the azimuth angle of the surface. The equation $\alpha$ relates the radiant intensity to the roughness of the reflective surface, with $\sigma_s$ being the standard deviation of the roughness of the surface and $k_s$ being determined by $\rho$ and $\theta_a$ which is the azimuth angle between the LED light and the surface of the color strip. The changes in the surface reflectiveness is due to the color of the strip at a specific point. As the beam travels over a section of the color strip the variations in the surface result in the optical power received by the sensor head changing. The received irradiance to the sensor head is found to be [15]

$$dE_{rr}(\theta_v,\theta_r,\theta_a)=\frac{dP_{rs}\left(\theta_v,\theta_r,\theta_a\right)}{\pi d_2^2}\cos\left(\theta_s\right) \tag{4.11}$$

where $\theta_s$ is the incident angle of the light reflected by the color strip with respect to the normal vector of the RGB sensing face.

With a known irradiance, it is possible to relate the output voltage of the sensor to a specific optical power using (4.4) and the relationship between the reverse-bias current

$$i_p=r_\Phi\Phi_e \tag{4.12}$$

where, $r_\Phi$ is the responsivity of the RGB sensor, and $\Phi_e$ is the radiant flux. The radiant flux can also be calculated as the surface area of the RGB photodiode times the received irradiance. Using the relationships described above it is possible to come to the final conclusion that [15]

$$v_o = r_e E_{rr}(\theta_v, \theta_r, \theta_a) \tag{4.13}$$

This equation can be used to directly calculate the change in output voltage with any change in the surface of the color strip.

## 4.3. Color Strip Specifications

The section of color strip used in these experiments is comprised of 500 unique sections of RGB intensity values of the color red, each with a width of 0.00035 m. The position of the platen in the *y*-axis is determined based upon the output of the sensor in response to a particular location above the color strip. A four centimeter section, about the initial position, from –1 cm to 3 cm was chosen to linearize. This section of the strip was chosen because it was within the working range of the laser interferometers, allowing for the laser interferometers to be used to collect the position data during runs using the RGB sensor.

## 4.4. Color Strip Resolution and System Limitations

The RGB sensor was placed 0.753 cm above the color strip creating a sensing length of 0.2659 cm at the level of the strip. This sensing length is significantly larger than the length of a single color intensity band, which has a length of 350 μm. The influence of multiple intensities of color being sampled simultaneously is an averaging of the values in the sensing area. This averaging effect allows for a theoretically infinite number of unique positions as any change in location will cause the output to vary. Since the sensor output is not limited to the number of individual variations of intensity values, the resolution can be improved beyond that of a single strip of color intensity, which is equal to 350 μm. The other limit, to the resolution of the color strip, is the quality of the printer used to create the color strip. The printer used for this research was an Aficio SP-C830DN color printer manufactured by RICOH with a maximum dots-per-inch (dpi) of 600, which equates to a potential resolution of 42 μm.

## 4.5. Linearization of Red Color Strip

In order to correlate the sensor output to position it was desired to have a simple linear response. The reason for this was to minimize the order of the curve-fit used as values of the integers calculated to determine the position are limited 8 bits, or a value of 65535, by the VME computer.

With red as the primary color the first run of the target area showed significant nonlinearities within the working range. In order to eliminate these nonlinearities the output of the RGB sensor is linearized using the inverse method. A program was developed for this process and can be found in Appendix A.2. The inverse method is a method by which a nonlinear signal that has a single unique output for every point sampled can be linearized by adjusting the initial model based on its output. Since the process uses the output to adjust the initial input it is termed the inverse method. The strip is linearized by taking the initial nonlinear output of the sensor and sampling it at the location of the center of each of the different sections of color intensity along the portion of the strip being linearized. These output values are then correlated with the intensity value of the red color strip around which they occur. The next step is to determine at which red intensity values the maximum and minimum sensor output occurred. The intensity values of these two points are used as the bounds for the linear strip, as well as to determine the slope value about which the intervening color intensities will be fit. An iterative process is then used to determine the intensity values of each of these intervening sections. At each section, the slope between the maximum value and each of the other intensity values is calculated, and compared to the slope determined in the prior step. The value that most closely fits this slope is assigned to the section. The process is then repeated until the strip is fully populated with intensity values. Through this optimization

process it is possible to filter out nonlinearities in the signal response. In order to optimize the linearization multiple iterations were needed to remove all nonlinearities from the output this was primarily due to the averaging effect, caused by multiple sections influencing the sensor response at any given location. By changing the values of adjacent sections, independent of one another, unfavorable interactions can occur in the response, which creates new nonlinearities. Limiting the inverse method to smaller subsections of the overall color strip, in which these small nonlinearities persist, can over several iterations be used to linearize the strip to a finer precision. Manual adjustment of intensity values may also be used to eliminate persistent nonlinearities.

The comparison between the original and linearized color strips can be seen in Figure 11. The sensor output of the two strips, in the desired operating range between –5 mm and 5 mm, can be seen in Figure 12 and 13. Initially there were significant nonlinearities in the signal. However, through the application of the inverse linearization method, the nonlinearities in the area of interest were significantly reduced. In order to better compare the responses, a moving average was used to limit the noise in the signal in order to more accurately compare the runs. In total, 19 iterations were performed to linearize the strip, with the best full run, –1 cm to 3 cm, iteration having a coefficient of determination ($R^2$) value of 0.9961. The majority of the nonlinearity of the output of this color strip exists at the ends of the working range, specifically between 0.026 m and 0.03 m. Since this section of the color strip is beyond the bounds of the runs, it was deemed an outlier. When excluded, the $R^2$ value of the color strip becomes 0.9986.

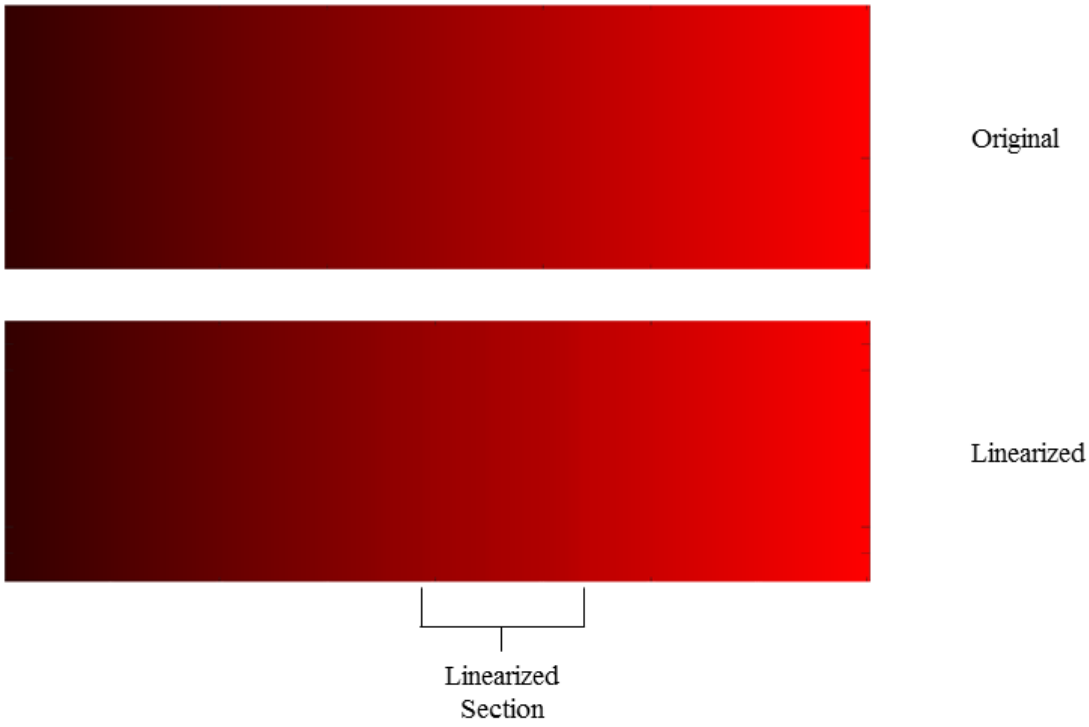



Figure 11: Original and linearized color strips

Additional linearization was done, with a focus on linearizing a length about the zero position of the RGB sensor. This additional linearization was done between −5 mm and 5 mm. After the linearization, the coefficient of determination between the signal output and the linear curve-fit was 0.9993. Because this length is significant and has an exceptionally high linearity, when compared to the rest of the strip, it was chosen to be the range in which the testing of the RGB sensor position control would be done.

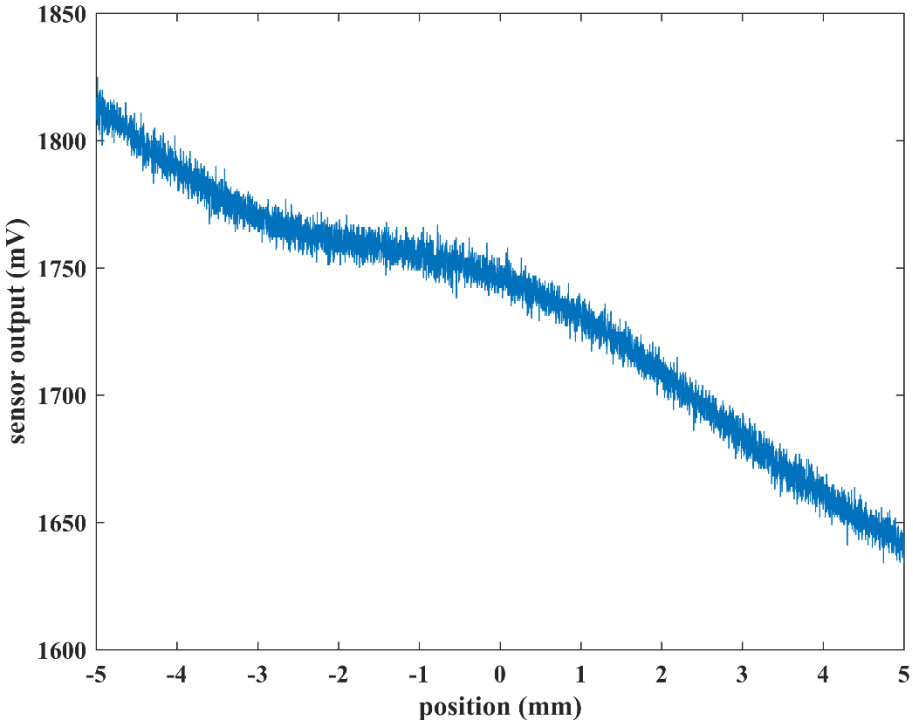



Figure 12: Initial sensor output of red color strip from −5 mm to 5 mm

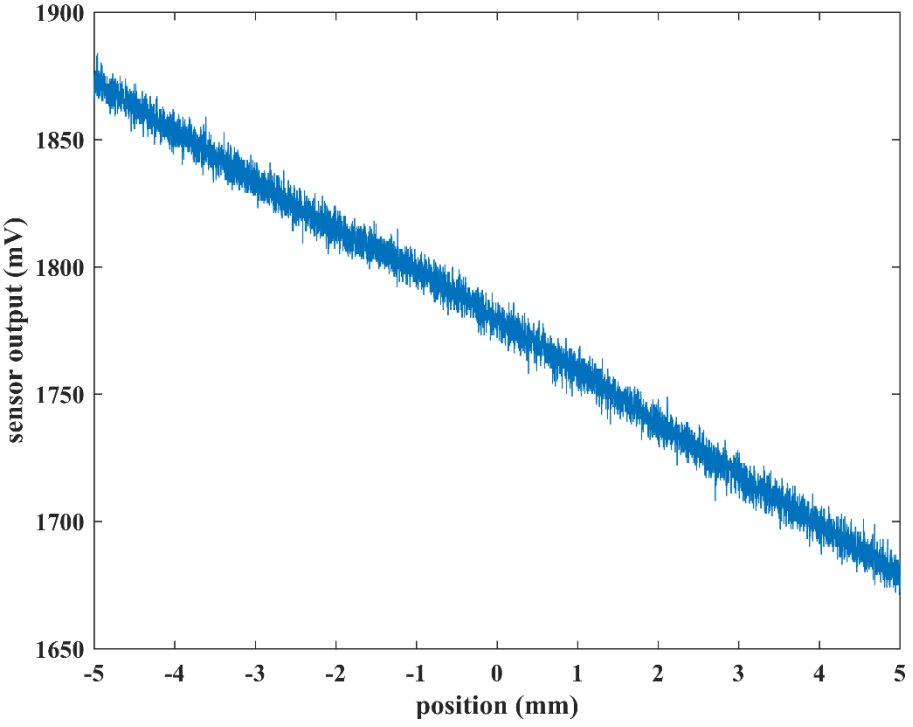



Figure 13: Sensor output of linearized red color strip from −5 mm to 5 mm

### 4.6. Blue Color Strip Linearization and Speculation of Multi-Color Interactions

The same linearization process was used to generate a linearized blue color strip. The purpose of linearizing the blue strip was to validate the linearization, as well as to determine if blue would be a viable color for use in RGB based position control systems. The linearized color strip and the corresponding output of the sensor with respect to the strip can be seen in Figures 14 and 15 below. The optimum linearized working area on the final blue color strip was found to be between 0 m and 5 mm.

The primary difference, seen between the output of the sensor in red and blue, is the voltage level of the output between the two colors. It is believed that because blue has such a significantly smaller output voltage when compared to red, that future research using both colors for use in 2-D controls is likely to be successful. The integration of a second color is likely to augment the output value of both the red and blue signal channels. Since the output ranges vary significantly, it is thought that there will be less coupling between the two colors allowing for more precise control in the two individual axes.

Figure 14: Linearized blue color strip

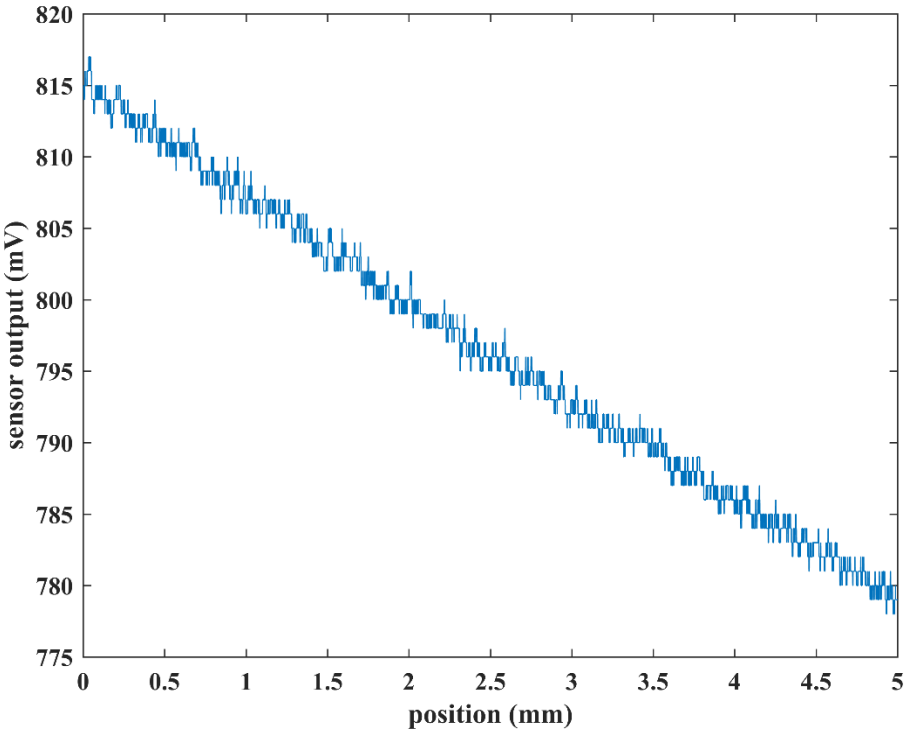
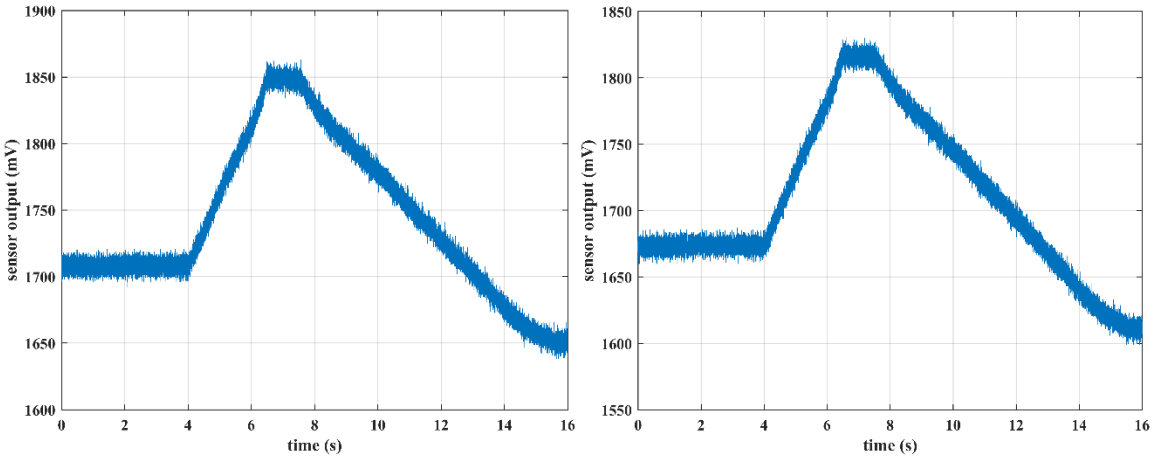



Figure 15: Linearized output of blue color strip from 0 m to 5 mm

### 4.7. Discussion of Control System Error Sources

The development of the controller and linearized color strip is important for creating a system that is able to provide highly accurate, repeatable movement using a feedback system that does not inherently lend itself to consistency. The inconsistencies in the system are believed to come from two possible places. First, the lack of contact with the global reference frame. Even with ideal conditions there is no way to completely remove the initial position error in the system. This error is primarily due to the Hall-effect sensors that have a resolution of 8–10 μm [17]. Since the Hall-effect sensors are used to set the initial position of the moving platen when the feedback loop in the *y*-axis is transferred to the RGB sensor, the sensor must adjust to the zero position of the color strip prior to any movement. In working on a scale of tens of micrometers this initial offset can negatively affect the accuracy of the position control. If the sensor cannot reach a steady-state value quickly, it will result in significant additional

noise. The second source of error, in the RGB based control system, is the sensor itself. It was found that over a series of runs that the sensor output at the initial position, found by the RGB sensor, varied by up to 100 mV. This is a significant amount, considering the desired resolution of this system. However, it was found that while the initial value of the RGB sensor varied, the distance per change in intensity value remained the same over a series of runs. To account for this difference in the output value an offset, which is determined based off of the difference in the initial sensor value at the start of a run and the zero position sensor value of the curve-fit, was applied.




(a) (b)

Figure 16: Side by side comparison of two calibration runs of the red color strip from 0.0 m to −0.01 m to 0.005 m

It can be seen from Figure 16 that the average initial value shifts between the two runs by approximately 30 mV. This same 30 mV offset can be seen in the maximum and minimum value of the sensor output. The addition of an offset calculated prior to each run is able to account for any fluctuation in the initial output from the RGB sensor, and thus remove the error in the position calculation that such a fluctuation would cause. The results shown in the next chapter will act as further validation to this statement.

# CHAPTER V

# CONTROLLER DESIGN

In this section, the design of the controller, used to control positioning in the *y*-axis using the feedback from the RGB sensor, as well as the implementation of said controller are discussed.

## 5.1. Control System Design

In order to account for the error sources discussed in Section 4.7, the control system must be robust enough to be able to quickly adjust to the errors, while still maintaining high position accuracy. The first controller structure considered was a proportional-integral-derivative (PID) type controller. Due to the naturally unstable nature of the maglev system, and the desire to minimize the response time of the system, neither proportional nor proportional-integral controllers was considered. Initially testing was done with a PID controller, with a high derivative gain in the system, the noise in the sensor would cause the derivative term to fluctuate drastically during a run causing the system to become unstable. A PID controller, with a low derivative gain, was not considered because there were other controller options that theoretically provide a faster response, with a minimal increase in complexity. More complex controller architectures, such as linear-quadratic (LQ) servo control, and fuzzy control methodologies were not considered for this system, because both lack a method to easily tune the system.

After other controller types were considered, it has been determined that a lead-proportional-integral (lead-PI) controller is optimal for this system. The lead-PI controller takes the form of

$$C_y(s) = K\left(K_p + \frac{K_I}{s}\right)\left(\frac{Ts+1}{T\alpha s+1}\right) \tag{5.1}$$

In the above equation $K$ is the gain of the overall system, $K_p$ is the proportional controller gain, $K_I$ is the integral controller gain, and $T$ and $\alpha$ depend on the phase to be added to the closed-loop system.

The proportional and integral gains of the PI controller allow for a smooth response, while also eliminating the steady-state error in the controller. At the same time the lead compensator improves the stability, by limiting oscillations in the closed-loop system as well as decreasing the maximum overshoot and improving the system response time. The combination of these two controller structures results in a system that is better set up to limit the effects of outside perturbations while maintaining a smooth, relatively fast response and remains easy to tune. The desired specifications, for the controller, were to eliminate steady-state error, while also minimizing the settling time and overshoot. The primary objective of these specifications was to facilitate multiple smooth and accurate movements in a single run, as would be seen in a step-and-scan type of motion. The dynamic model of the platen in the *y*-axis has the form of

$$G_y(s) = \left(\frac{1.8888}{s^2}\right) \tag{5.2}$$

This model was developed by substituting the left side of (3.17) into (3.15) and then substituting the left side of (3.15) into the left side of equations (3.1)–(3.6) to form [17]

$$\begin{bmatrix} m\ddot{x} & m\ddot{y} & I_z\ddot{\theta}_z & m\ddot{z} & I_x\ddot{\theta}_x & I_y\ddot{\theta}_y \end{bmatrix}^T = 2b_*(z_0)\begin{bmatrix} u_x & u_y & u_{\theta z} & u_z & u_{\theta x} & u_{\theta y} \end{bmatrix}^T \quad (5.3)$$

This leads to the transfer function, in the *y*-axis, being

$$G_y(s) = \frac{2b_*(z_0)}{ms^2} \quad (5.4)$$

With a mass of 0.9 kg, and with the value for $b_*(z_0)$ found in Chapter III of 0.85, the dynamic model of the platen, shown in (5.2), was derived.

Once the appropriate control structure had been decided, the continuous-time lead-PI controller was designed. To achieve a smooth response with limited oscillations it was determined that the amount of phase added to the system should be maximized. The result of the large added phase is the pole of the lead compensator is moved significantly far away from the poles and zeroes of the PI controller and the plant, which leads to a nearly flat sensor response with an initial overshoot, but no oscillation. The primary reaction of the system is dominated by the lead compensator, however the proportional and integral gains in the lead-PI controller are also essential to the response. The integral term allows for additional limiting of the settling time of the response, and the zero added due to the proportional gain is essential to system stability. The drawback to a large *K* is that there is an increase in the high-frequency noise passed through to the system. The *K* value for the controller was determined through a series of iterations to find the value that produced the best response, with the least amount of high-frequency noise. The final iteration of the controller used for testing is given below in two different forms.

$$C_y(s) = 10735.53\left(1.0 + \frac{1.4}{s}\right)\left(\frac{s + 15.3374}{s + 299.401}\right) \tag{5.5}$$

$$C_y(s) = 500\left(1.0 + \frac{1.4}{s}\right)\left(\frac{0.0652s + 1}{0.00334s + 1}\right) \tag{5.6}$$

In order to determine the stability of the controller, the root locus of the loop transfer function was drawn, and can be seen in Figure 17. At a *K* value of less than 9.3, the system is unstable. However, the gain used in the controller of 550 is considerably large, and in a stable region of the Root locus.

Figures 18 and 19 show the Bode plots of the loop transfer function and the closed-loop transfer function, with the controller shown in (5.5) and (5.6). The phase margin is 63.3°, and a crossover frequency of 67.8 rad/s. The closed-loop bandwidth of the controller was found to be 14 Hz.

The response characteristics of the controller in (5.5), during a 1 mm step response, can be seen in Figure 23. In order to show the response characteristics as clearly as possible the laser interferometers where used to close the feedback loop. The response has zero steady-state error, as well as zero overshoot, and a rise time of 2.25 s. This response satisfies all of the desired specifications for the controller.

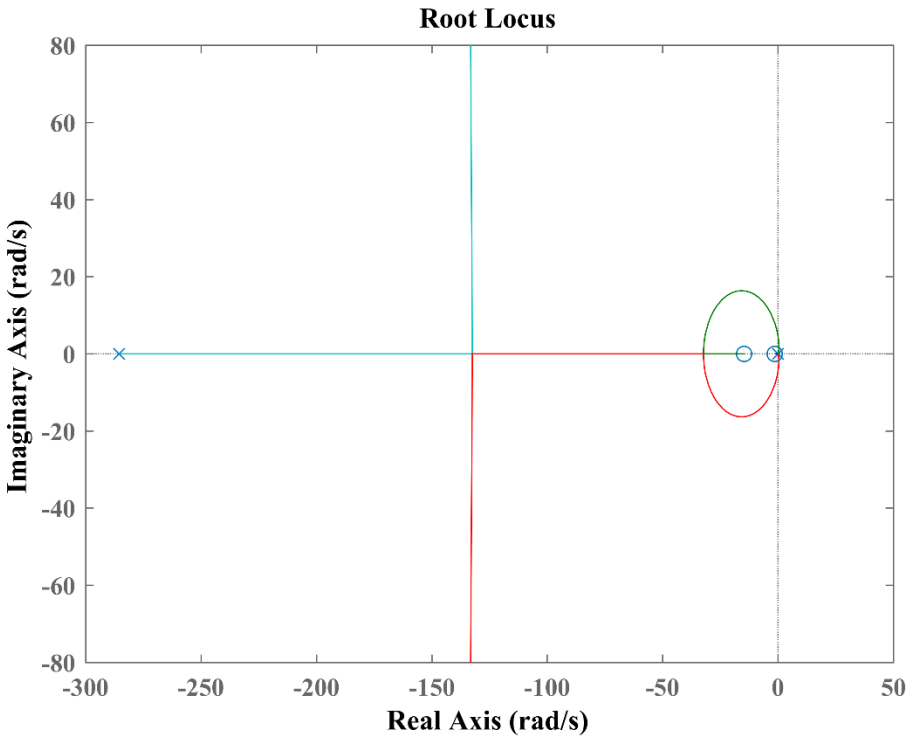



Figure 17: Root locus of controller in (5.5)

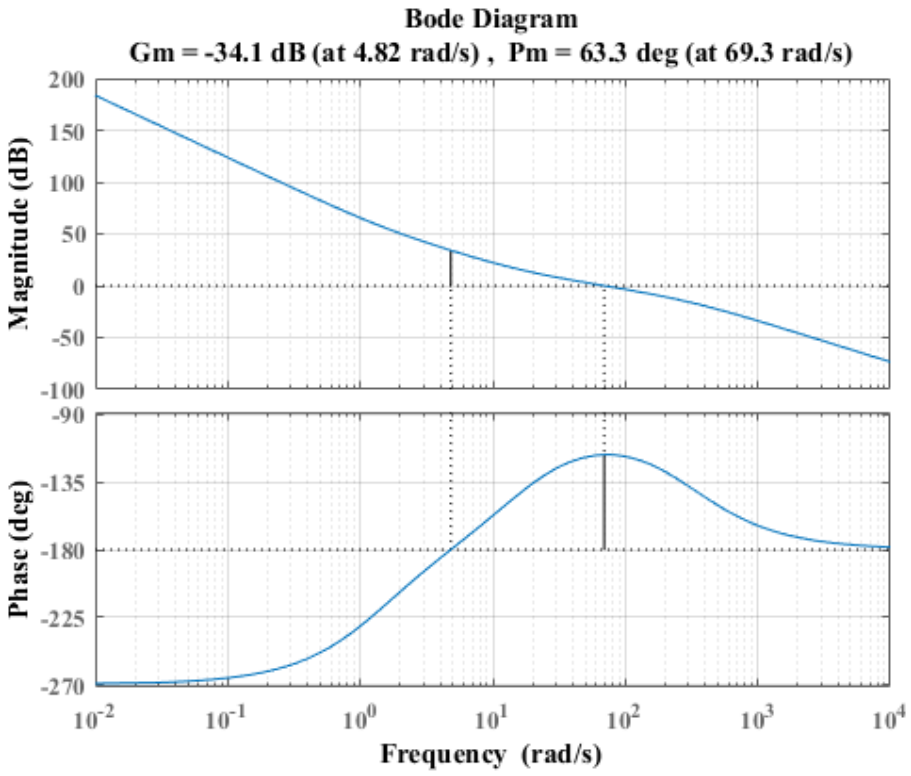



Figure 18: Bode plot of the loop transfer function of controller in (5.5)

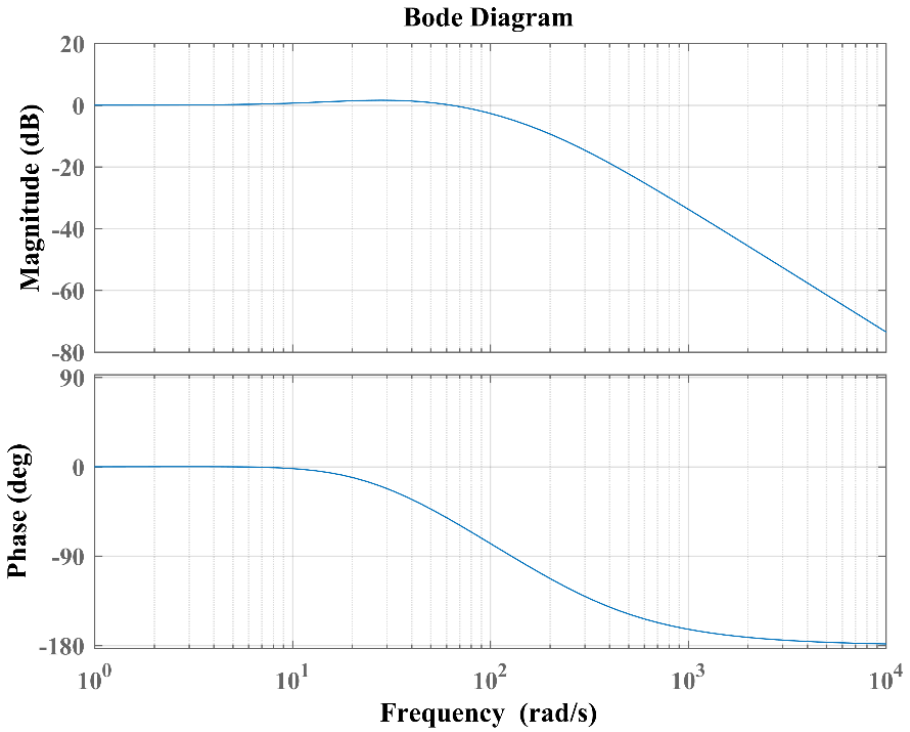



Figure 19: Bode plot of the closed-loop system of controller in (5.5)

In order to fully describe the capabilities of the RGB sensor in position control another controller allowing for overshoot, was implemented. The purpose for integrating this controller is to examine the effects of allowing overshoot on the rise and settling time of the maglev system. The continuous-time controller that provided the best response with overshoot was found to be

$$C_y(s) = \frac{200(0.994)}{0.007136}\left(2+\frac{5}{s}\right)\left(\frac{s+10.0603}{s+140.1345}\right) \tag{5.7}$$

Figures 20–22 show the plots used to describe controller (5.7). Figure 20 is the root locus of the loop transfer function of the controller in (5.7). At a *K* value of less than 5.9, the system is unstable. However, the gain used in the controller of 200 is considerably large, and in a stable region of the Root locus. Figures 21 and 22 show the Bode plots of the loop transfer

function and the closed-loop transfer function, with the controller shown in (5.5) and (5.6). The phase margin is 53.5° and a crossover frequency of 68.3 rad/s. The closed-loop bandwidth of the controller was found to be 18 Hz.

When comparing the controller in (5.7) with the controller in (5.5) it can be seen that the controller in (5.8) has a larger proportional and integral gain, as well as a decreased distance between the pole and zero in the lead compensator. The result of the decrease in the distance between the zero and pole along with the increase in the proportional and integral gains is a quicker response, as seen in Figure 24, which shows a 1 mm step responses using controller (5.7). The step response in Figure 24 has a rise time of approximately 0.3 s compared to a rise time of 1.35 s for the controller in (5.5), presented in Figure 23. Both responses use the laser interferometers to close the feedback loop. The shorter rise time comes at the expense of a significantly increased settling time of the controller in (5.7).

In order to limit the settling time the distance between the pole and zero, of the lead compensator, was decreased. This decrease leads to more oscillation in the response initially, but a shorter settling time. The time to reach a steady-state value for controller (5.7) is 5.4 s compared to 2.9 s for controller (5.5). The percent overshoot of controller (5.7) is 5.0%. The response of controller (5.7), when the RGB sensor is used to close the feedback loop, will be presented in Chapter VI.

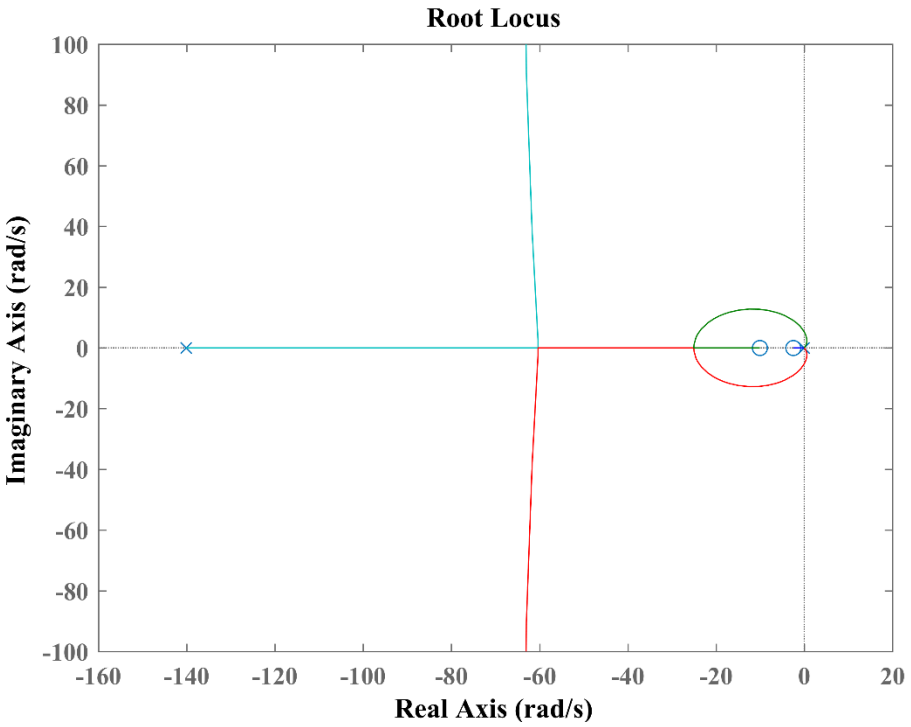



Figure 20: Root locus of controller in (5.7)

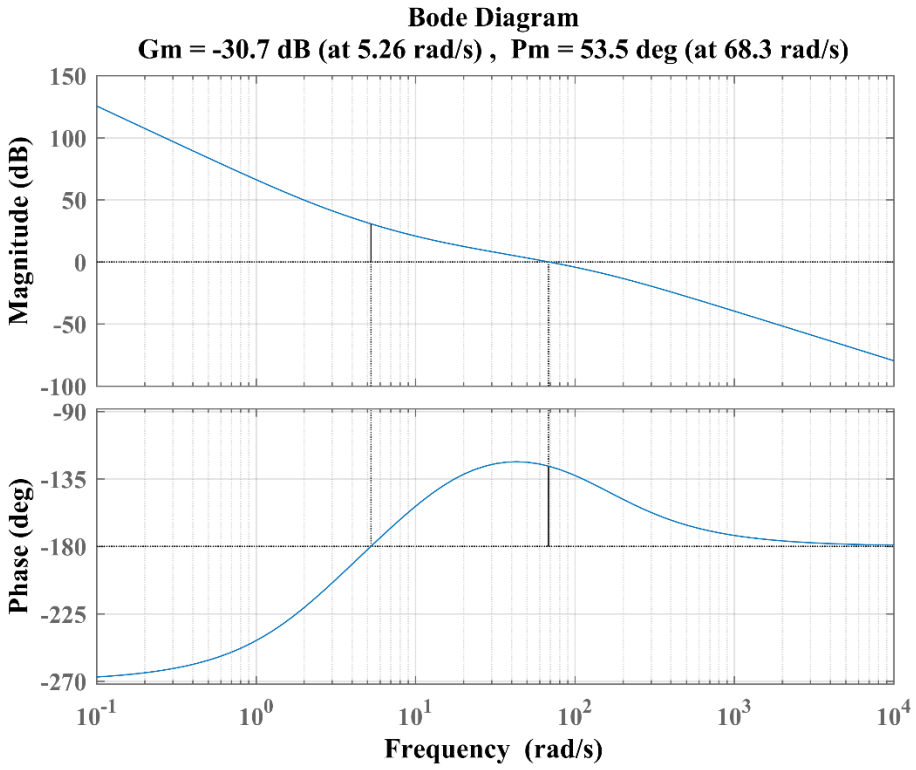



Figure 21: Loop transfer function of controller in (5.7)

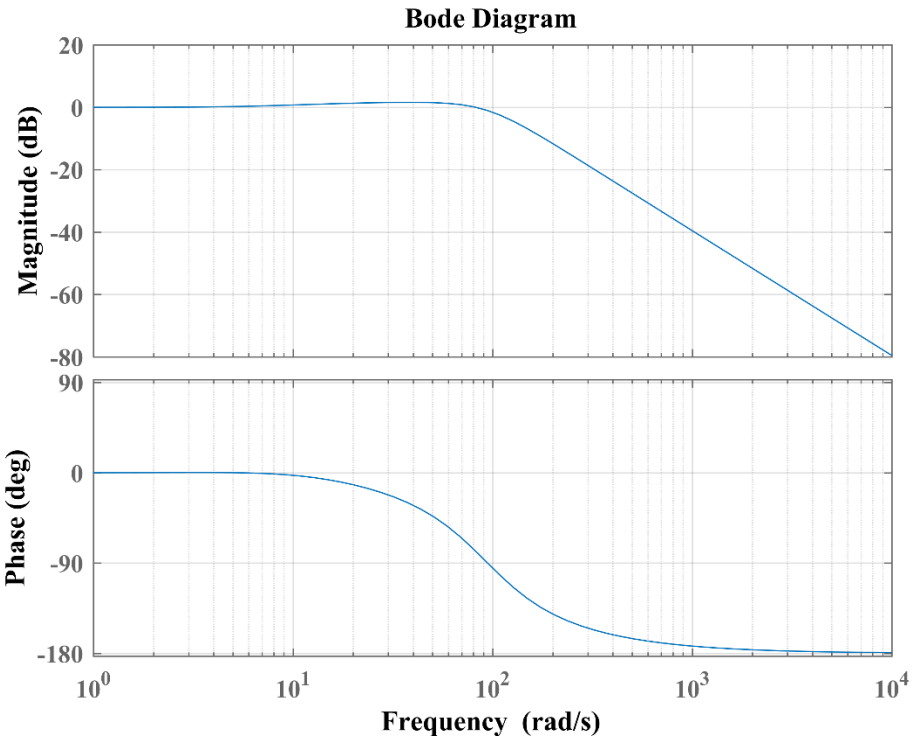



Figure 22: Closed-loop transfer function of controller in (5.7)

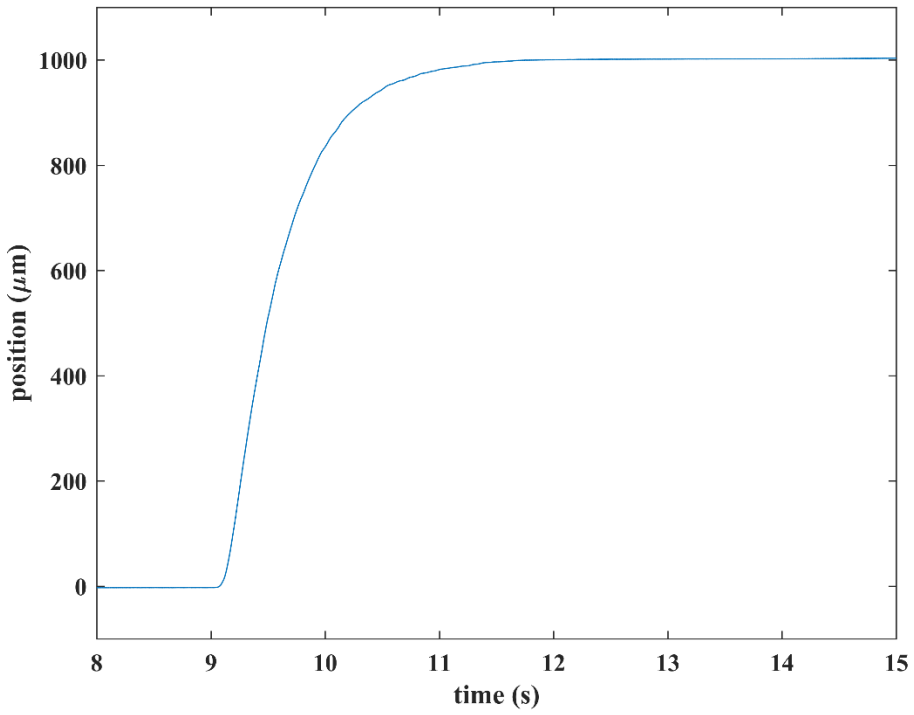



Figure 23: 1-mm step response of controller in (5.5) using laser interferometers

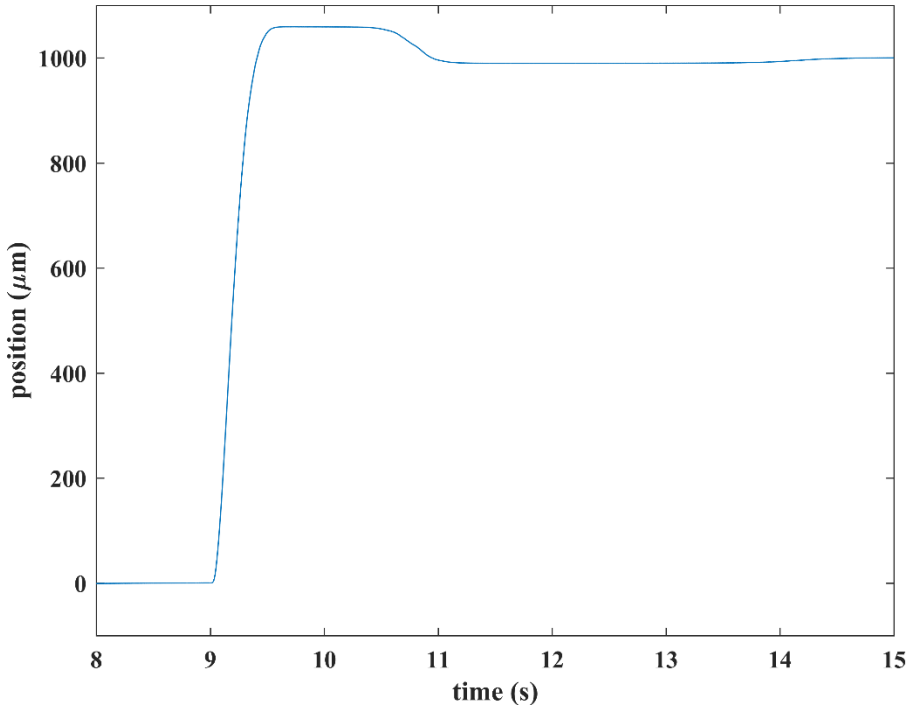



Figure 24: 1-mm step response of controller in (5.7) using laser interferometers

## 5.2. Implementation of Controller

In order for the maglev system to function properly, a series of actions must occur over the course of a run. Initially, prior to starting the system, a zero position for the moving platen must be determined. This position must be centered within the magnetic array to maximize the possible area for movement to occur. When the initial position is determined, the platen must be held steady while the laser interferometers are aligned with the precision mirrors attached to the platen. Once the laser interferometers are aligned with the platen, and the receivers, it is then possible to begin the startup procedure for the moving platen. This procedure is presented in Figure 25.

At the beginning of a run, the first step is to align the platen with the initial position. At this initial position the HeNe laser beam from the laser head and among all of the laser interferometers are aligned so that all three of the receivers for the laser interferometers are

active. Once the system is activated, there is a period of time allowed for adjustments of the platen, in order to improve the initial alignment before the feedback control begins.

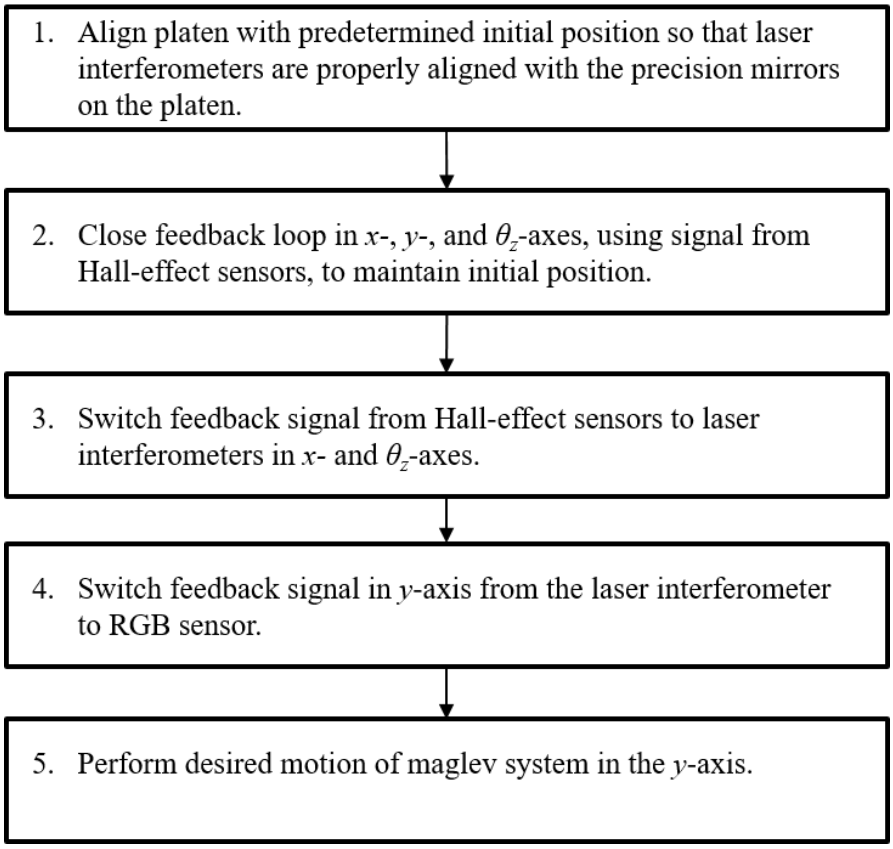



Figure 25: Setup method for maglev system

In step 2 of Figure 25, the two Hall-effect sensors located on the platen are used to close the feedback loop of the controller in the $x$-, $y$-, and $\theta_z$-axes. As long as the platen is placed within a few millimeters from the initial position, the platen can automatically find and hold its initial position by using the Hall-effect sensors. While the Hall-effect sensors are maintaining the platen's initial position, the laser interferometers should be aligned with their respective receivers, and the signals should be received by the laser axis boards on the VME chassis. The position resolution of the Hall-effect sensors, of 8–10 μm, is small enough to allow for this position initialization to be accomplished. The zero position of the laser

interferometers is set once the feedback control of the Hall-effect sensors has found the initial position. The zero position of the laser interferometers is the first position captured after the laser interferometer sensors are enabled to start acquiring data.

In the 3rd step, the feedback signal is switched from the Hall-effect sensors to the laser interferometers for the $x$- and $\theta_z$-axes. The platen is then allowed a short period of time to settle, before the feedback signal in the $y$-axis is switched from the Hall-effect sensors to the signal from the RGB sensor. Due to the position of the RGB sensor being based off of the linearized color strip, the initial position from the Hall-effect sensors and laser interferometers varies from that of the RGB sensor. The result of this variance is a perturbation in the platen at the time the feedback signal is switched. This perturbation can vary in size depending on the location of both zero positions. In order to account for this perturbation, the platen is given four seconds to settle prior to any movement of the platen.

Once the above steps are complete, control input can be applied and the desired motion can be completed. Due to the use of air bearings, to keep the platen at a constant position in the $z$-axis during the experiments, the control of the $\theta_x$-, $\theta_y$-, and $z$-axes can be ignored. The real-time calculations of the position for the Hall-effect sensors, laser interferometers, and RGB sensors are done in an interrupt service routine (ISR) in the VME computer. In the ISR the appropriate conversion of the sensor signals are completed, any relative position calculations are done, error calculations are completed, and the necessary control input is determined and converted to the appropriate current value to be sent to the coils. The sampling frequency of 4 kHz means that the ISR must be completed within 250 μs. The available time for the ISR is sufficient, as a test program with a comparable number of operations was able to run within 100 μs [17].

The controller used in this thesis was designed as a continuous-time controller in MATLAB. This was done in order to better tune the predicted response, using the closed-loop system's Bode plot and step response. For implementation into the real-time control system it was necessary to convert the continuous-time controller into a discrete one. A zero-order-hold (ZOH) method was used to complete this task. This method was chosen because it assumes the inputs are piecewise constants at each sample time [22], which will work for this system, without any losses due to sampling between the continuous and discrete time controllers. Due to the complexity of the ZOH transformation, and the high sampling frequency of the system, both Tustin's and matched-pole-zero (MPZ) methods were considered as well.

To implement the continuous-time controller, shown in (5.8), into the maglev control code it was put in the form of a difference equation.

$$u_y[k] = 1.9279u_y[k-1] - 0.9279u_y[k-2] + 10742e_y[k] - 21440.985e_y[k-1] + 10699e_y[k-2] \tag{5.8}$$

# CHAPTER VI

# EXPERIMENTAL RESULTS AND DISCUSSION

## 6.1. RGB-Sensor Calibration and Color-Strip Linearization Verification

As discussed in Chapter II, the position data during the experiment were recorded using the redundant laser interferometers in the *y*-axis. In order to show that the values recorded using this method accurately represent those obtained with the RGB sensor and linearized color strip, it is necessary to compare the responses. Figures 26 and 27 show the same 1-mm step response in the *y*-axis with the control loop closed using the RGB sensor. It can be seen that the position values calculated from the RGB sensor output match those from the laser interferometer with a larger amplitude of noise, which was anticipated. This result verifies the accuracy of the sensor calibration and the linearization procedure used to create the color strip. In Figures 26 and 27 the 1-mm step responses have a rise time of 1.35 s with no overshoot, which is the same as the rise time when the laser interferometers are used to close the feedback look as seen in Figure 23.

## 6.2. Responses of Various Step Sizes

Various responses were taken using the red linearized color strip. The steps in each of the figures were commanded at 9 s. In Figure 28 a step response of 500 μm occurs. The response has a 0% overshoot and a rise time of 1.15 s. Figure 29 shows a step response of 100 μm. The step response has a rise time of 0.8 s and 0% overshoot. Both figures have a peak-to-peak noise amplitude of approximately 200 μm.

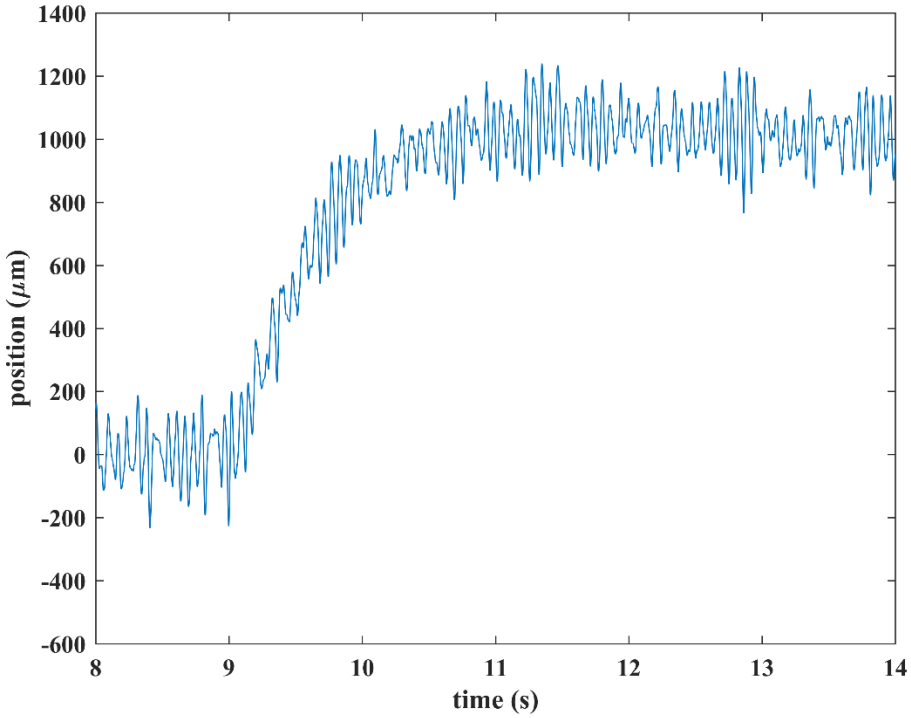



Figure 26: 1-mm step response using RGB with position data from laser interferometers

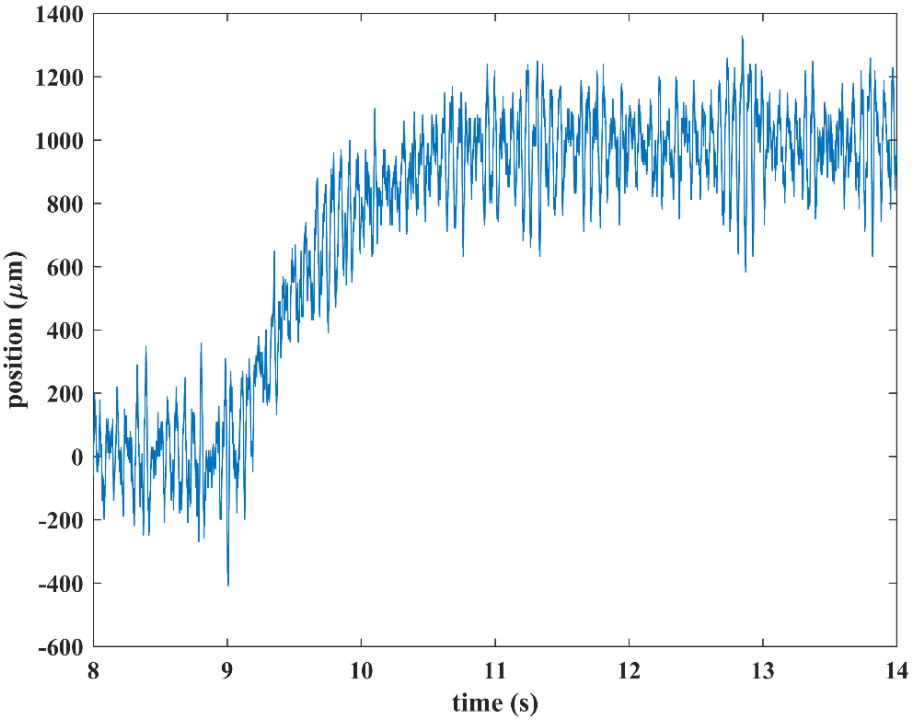



Figure 27: Calculated position using RGB and linearized color strip during a 1-mm step response

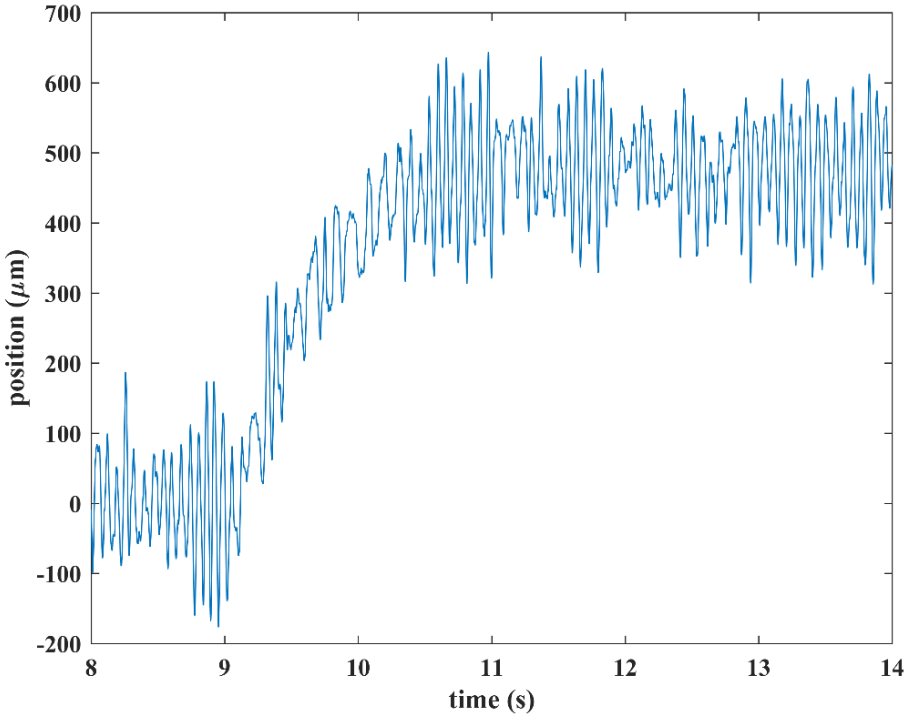



Figure 28: 500-μm step response

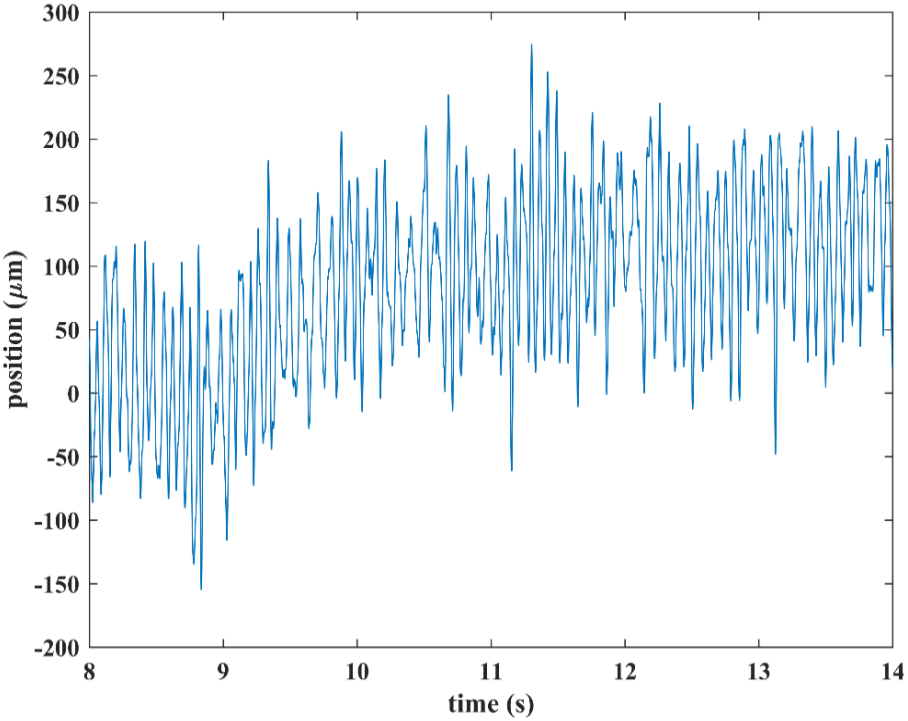



Figure 29: 100-μm step response

At less than 100 µm, the step responses become more convoluted as the noise amplitude of the output is greater than the size of the step. In order to show the overall trend of the response a 60-µm step, shown in Figure 30, has been smoothed using a local regression algorithm as shown in Figure 31. The MATLAB function smooth was used to apply the local regression algorithm, the function uses a weighted linear least squares and a 2nd-order polynomial to remove the noise from the signal. The local regression, used to produce Figure 31, uses a span of 10% of the data points to calculate the response. The downside of this method of smoothing is that, the regression algorithm is truncated at the ends of the dataset, which inaccurately skews the data points calculated within 5%, or 0.39 s, of the end of the plot. The 60 µm step response, shown in Figure 30 has a rise time of 0.75 s and a peak-to-peak noise amplitude of approximately 250 µm.

The position resolution of the closed-loop control system with the RGB sensor was found to be 30 µm and can be seen in Figure 32. Figure 33 shows a plot of the 30-µm step response that has been smoothed, using the same local regression as in Figure 31. The step was commanded at 9 s. The noise amplitude of the position is approximately 250 µm. This 30-µm step response demonstrates that the averaging effect, discussed in Chapter IV, does in fact hold true. With the color strip having a minimum resolution of 42 µm, due to the 600-dpi resolution of the printer, a step response of less than this value can only occur from the interactions between multiple adjacent color intensity values being sampled simultaneously.

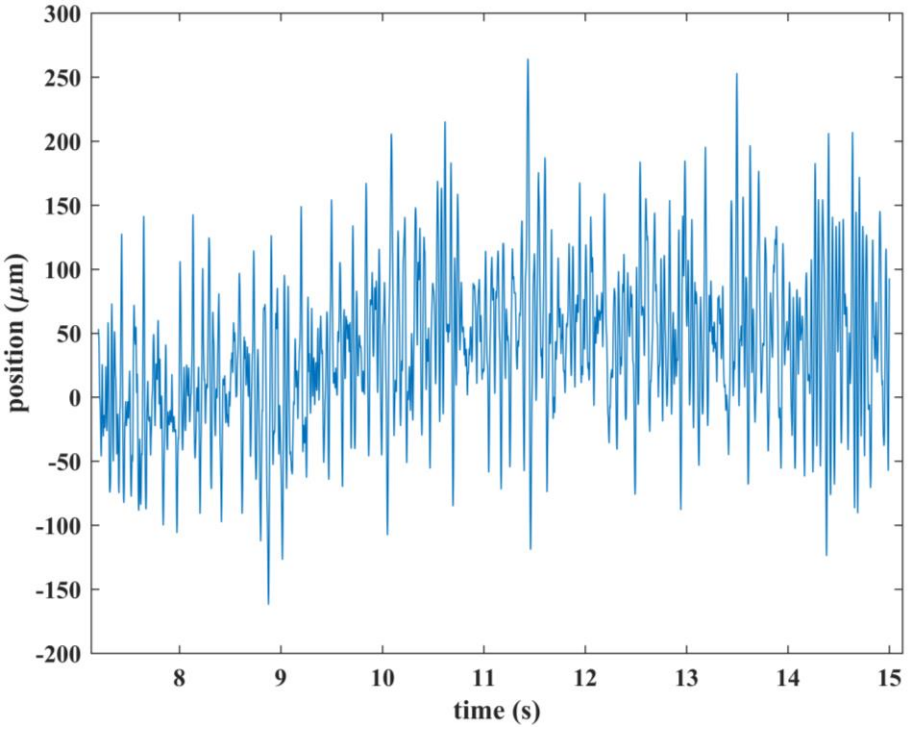



Figure 30: 60-μm step response

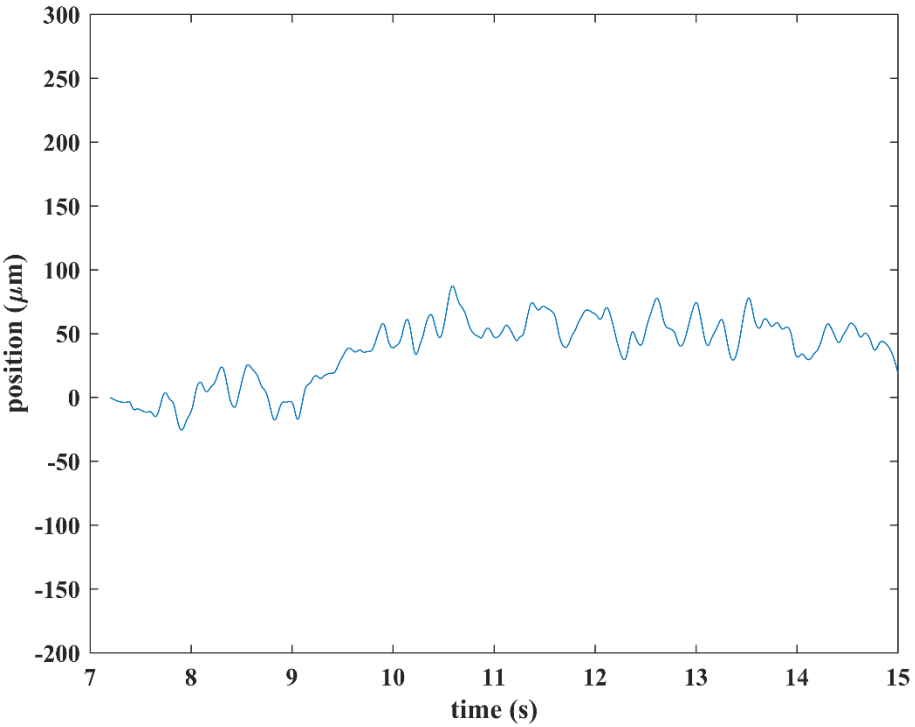



Figure 31: Smoothed 60-μm step response

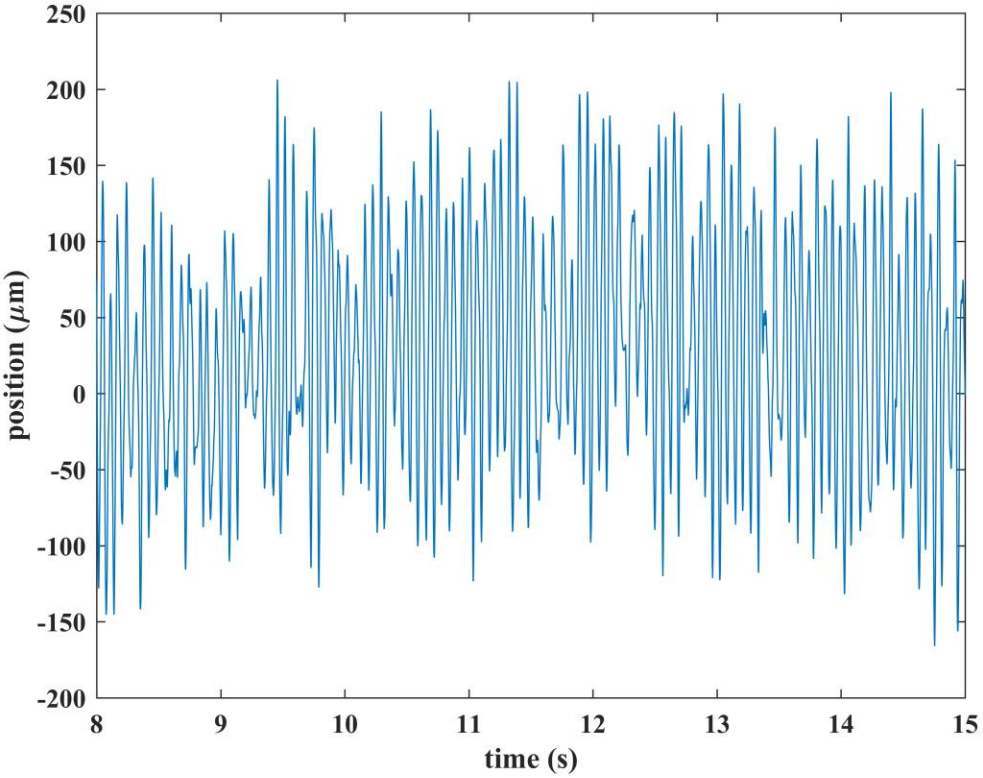



Figure 32: 30-µm step response

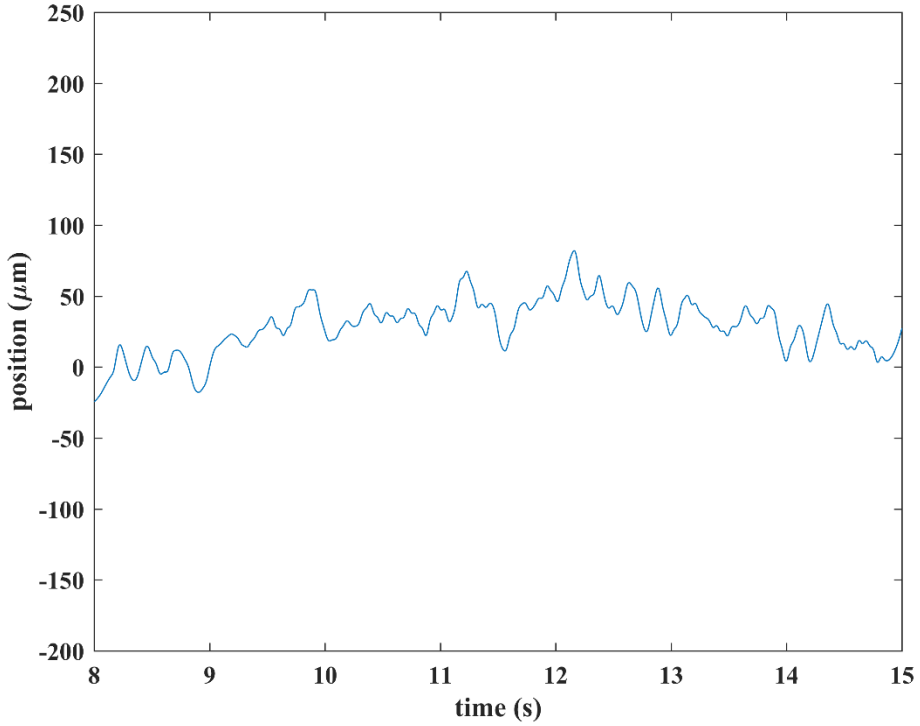



Figure 33: Smoothed 30-µm step response

## 6.3. Controller Comparison

When the closed-loop control system for the RGB sensor was developed, two different controllers were found to produce reasonable results with zero overshoot. They are given as follows.

$$C_y(s) = 10735.53\left(1.0 + \frac{1.4}{s}\right)\left(\frac{s + 15.3374}{s + 299.401}\right) \tag{5.5}$$

$$C_y(s) = \frac{200(0.0853)}{0.01125}\left(1.2 + \frac{2}{s}\right)\left(\frac{s + 11.7233}{s + 88.888}\right) \tag{6.1}$$

The controller in (5.5) is the final iteration of the controller used to produce the results presented in Section 6.2. Its development was discussed fully in Section 5.2. With a large distance between the pole and zero in the lead compensator, the system response has a short rise time as well as minimal overshoot. The large gain, however, does allow for an increase in the noise amplitude and frequency, when compared to the controller in (6.1).

The controller in (6.1) was designed with a focus on minimizing the noise amplitude frequency in the step response. To that effect the controller in (6.1) has a smaller gain value than controller (5.5), at 200, as well as less added phase, 50° compared to 65°. The result of this smaller gain is a lower closed-loop bandwidth of 10 Hz as shown in Figure 35. The decrease in bandwidth results in step responses with longer rise and settling times, but with a decrease in the output noise when compared to controller in (5.5). The proportional and integral gains were determined through a series of design iterations to determine the appropriate values for the maglev system. The final version of controller in (6.1) has a phase margin of 47.1° at crossover frequency of 37.4 rad/s. The Bode plots and the root locus for controller in (6.1) can be seen in Figures 34–36.

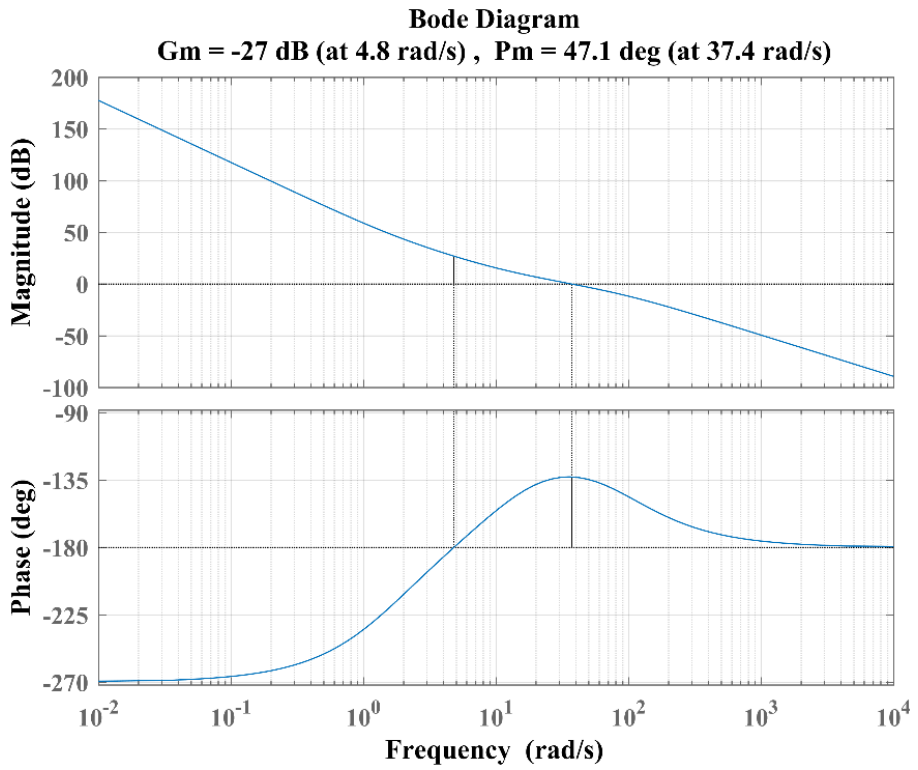



Figure 34: Bode plot of loop-transfer function of controller in (6.1)

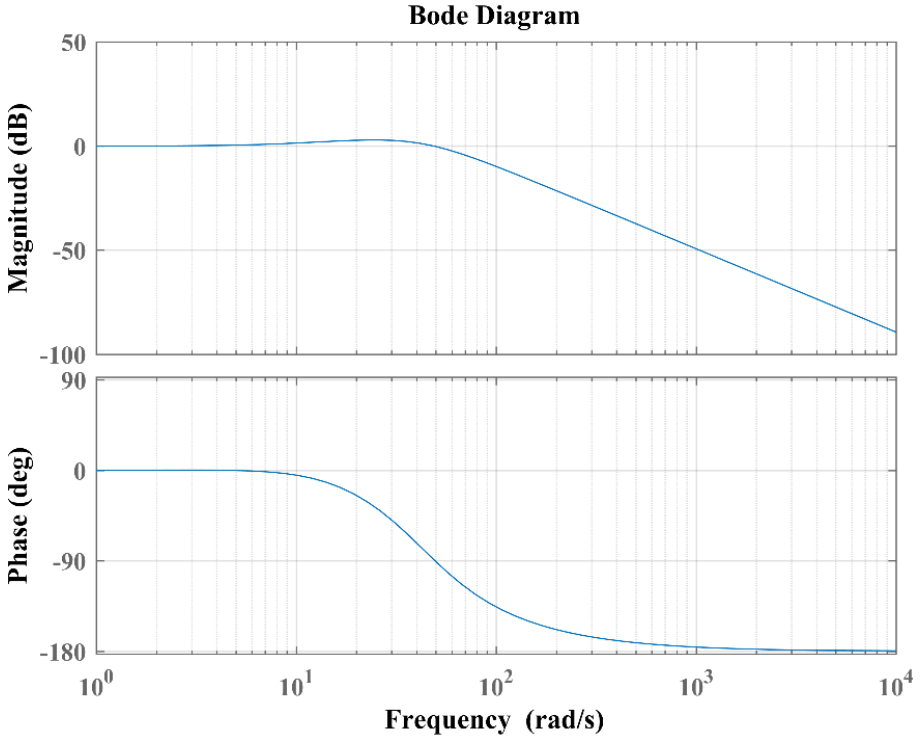



Figure 35: Bode plot of closed-loop transfer function of controller in (6.1)

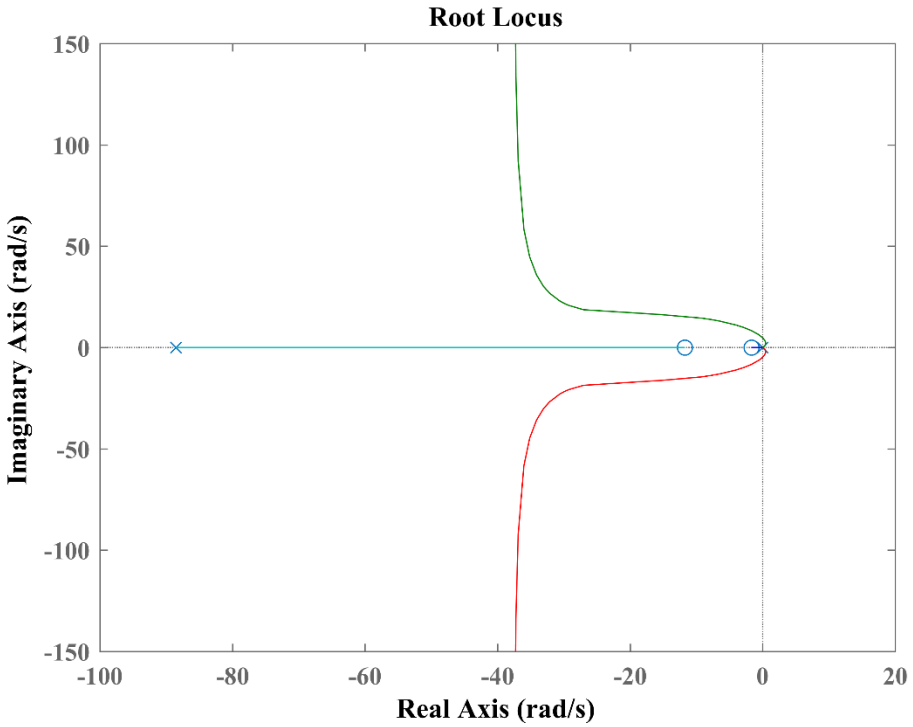
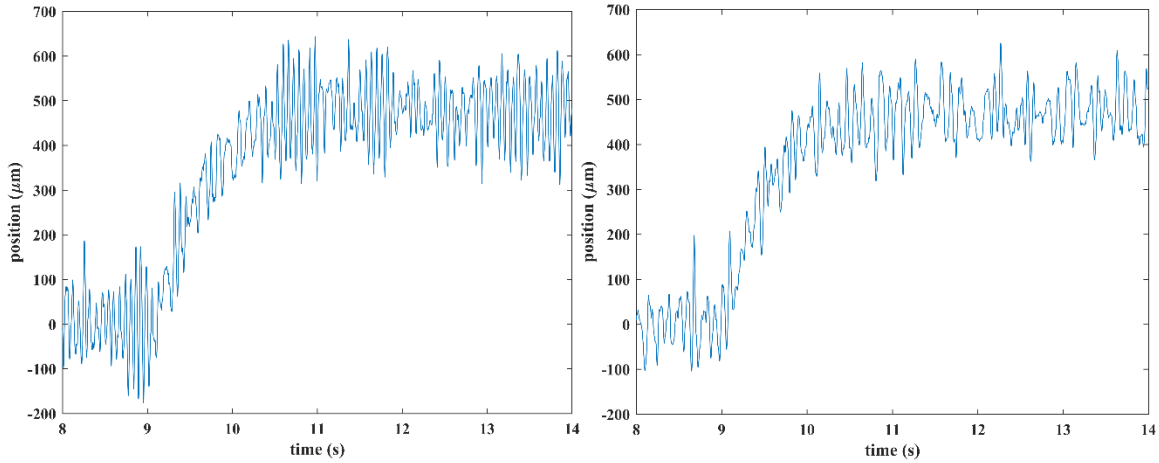



Figure 36: Root locus of controller in (6.1)

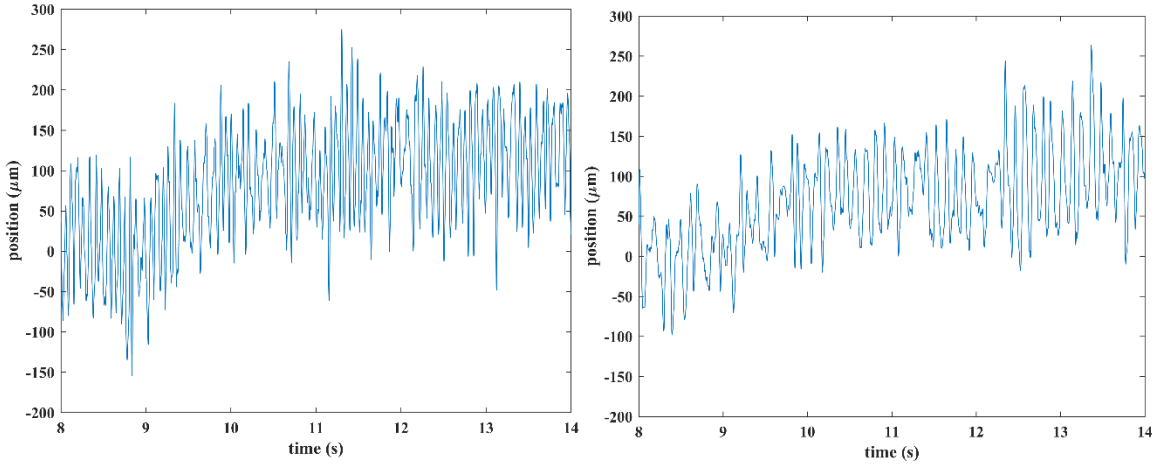
The 500-µm step responses, shown in Figure 37, are the most similar between the two controllers. The rise time of 1.2 s, shown in Figure 37(a), is the same as that shown in Figure 37(b), which has a rise time of 1.2 s. The rise time of each of the responses was found by using the time for the signal to go from 10% to 90% of the step response. Due to the noise in the signal the two bounds were determined as the time at which the average value of the signal response most closely represented the 10% and 90% points of the response. At this step size the controllers behaved similarly, however, it will be seen that at smaller step sizes controller (5.5) becomes significantly faster. This shorter rise time was expected from the increase in the gain, and added phase, of controller (5.5). The inverse relation in the bandwidth and the amplitude of the noise between the two signals was also seen. The peak-to-peak noise amplitude, in Figure 37(a), was approximately 250 µm, while the approximate peak-to-peak noise amplitude in Figure 37(b) was 200 µm. The same amplitude trends in the 500-µm step

response were also present in the 100-µm response, shown in Figure 38. The rise time of controller (5.5), shown in Figure 38(a), was 1.0 s compared to that of controller (6.1), shown in Figure 38(b), which had a rise time of 2.0 s.




(a) (b)

Figure 37: (a) 500-µm step response using controller in (5.5), (b) 500-µm step response using controller in (6.1)




(a) (b)

Figure 38: (a) 100-µm step response using controller in (5.5), (b) 100-µm step response using controller in (6.1)

The significantly larger rise and settling time of the lower gain controller in (6.1) was the primary reason it was not perused as the final version of the controller for this system. With the step sizes smaller than 100 µm the system response was not fast enough to find an accurate steady-state value in the limited time the controller has to run. This would lead to a response where the step could not be identified.

It is also necessary to compare the response of the controller in (5.5) to the controller in (5.7), which was described in Section 5.2. As discussed in Section 5.2 the controller in (5.7) was designed to allow for overshoot in the system response, leading to the rise time being significantly smaller than that found using the controller in (5.5). Inversely, the settling time of the system was significantly increased, compared to controller in (5.5). Figures 39 and 40 show a 500-µm and 100-µm step response using controller (5.7). The rise time of the two responses is 0.29 s and 0.55 s, respectively. The 500-µm step response, shown in Figure 39, has approximately 8% overshoot and a settling time of approximately 2 s. The 100-µm step response, shown in Figure 40, has approximately 20% overshoot and a settling time of approximately 2 s.

Despite the significantly shorter rise time of controller (5.7), the time to reach the steady-state is longer than that of controller (5.5). For a 500-µm step response the time between when the step is commanded and when the steady-state value is reached for controller (5.5) is approximately 1.5 s. Comparatively, controller (5.7) takes approximately 2.0 s to reach a steady-state value. Despite the significantly shorter rise time of controller (5.7), it is believed that controller (5.5) is still a better option for controlling the maglev system as this thesis is focused around precision motion.

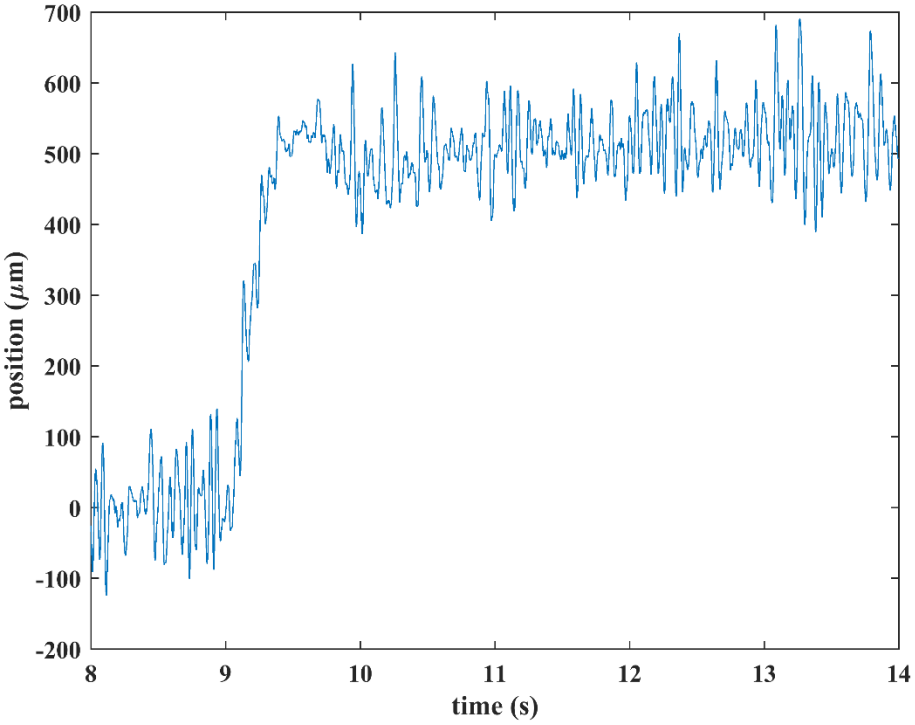



Figure 39: 500-μm step response using controller in (5.7)

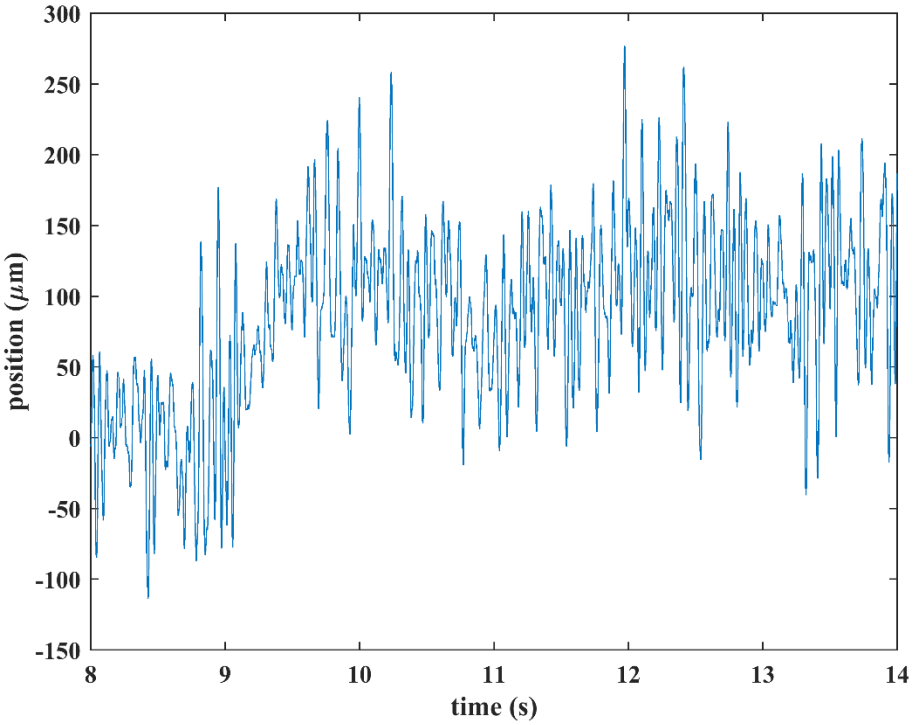



Figure 40: 100-μm step response using controller in (5.7)

## 6.4. Blue Color-Strip Step Response

In order to show the applicability of the RGB sensor in position control, the linearization process was repeated with a blue color strip, as discussed in Chapter IV. Both 500-µm and 100-µm step responses with the controller in (5.5) are presented in Figures 41 and 42, respectively. The 500-µm step response was found to have a rise time of 1.2 s. The 100-µm step response was found to have a rise time of 1.4 s, and a steady state error of approximately 20 µm. The blue color strip had similar response characteristics to that of the red color strip. However, the noise amplitude of the blue output is higher with an average peak-to-peak noise of approximately 300 µm. This increased noise amplitude causes the response of the system to be less accurate, leading to the larger steady-state error. The increased noise in the position response is due to the sensor output of the blue channel. Over the range of movement of the RGB sensor, from −5 mm to 5mm, the swing in the blue output is 35 mV, as shown in Chapter IV. Conversely, the swing in the red output over the same range is 200 mV. This smaller sensor output swing worsens the resolution of the sensor, which leads to the increased noise in the blue sensor. The noise amplitude of the blue output could be decreased by finding a range of blue RGB values through which there is a greater voltage change per distance or modifying the current strip with higher intensity strips of blue in the working area to augment the response.

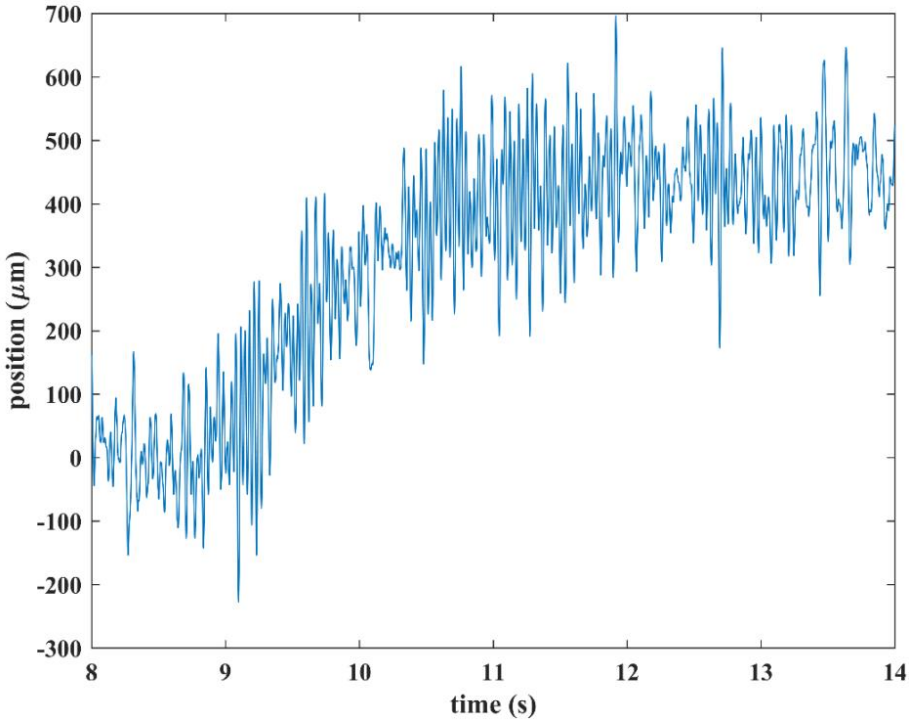



Figure 41: 500-μm step response using blue color strip

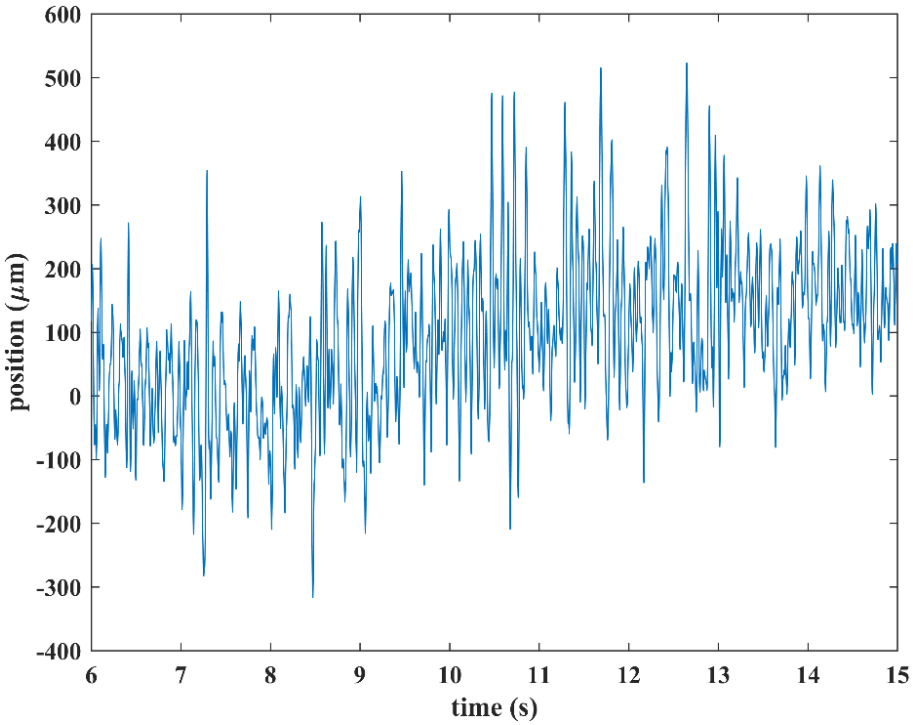



Figure 42: 100-μm step response using blue color strip

## 6.5. Step-and-Scan and Other Continuous Motion Profiles

Figures 43 and 44 present a trapezoidal response between the origin and 1 mm with the controller in (5.5). This trajectory models a common motion used in industry, referred to as a step-and-scan motion. The trapezoidal motion is shown using both the red and blue linearized color strips. Their time responses are about the same with the increased noise in blue as explained in the previous section.

Figure 45 and 46 show a series of step responses completed during a single run. Figure 45 is a set of two consecutive 100-µm steps, with the first step occurring at 9 s and the second step occurring at 12 s. Figure 46 shows a set of two step responses, the first from the origin to –100 µm, and the second from –100 µm to 100 µm. These steps occurred at 9 s and 12 s, respectively. Both sets of motion profiles highlight the capabilities of the RGB sensor in being able to complete precession motion responses in an accurate and repeatable manner.

Figure 47 shows a continuous motion response from the origin of the red color strip to 1 mm, using a step size of 25 nm. The step response was commanded at 6 s. The system is controlled by the laser interferometers until 3.6 s. At the time the controllers are switched, and the platen readjusts to the origin of the color strip coordinate frame. Because the RGB sensor has a relative coordinate system, which is not associated to that of the laser interferometers, the zero position of the two sensors are different. Due to the position being controlled by the RGB sensor, but recorded using the laser interferometers, there is an offset in the position. This is the reason why Figure 47 is shifted up by approximately 240 µm. The response of the system shows that the control system is capable of fine movement profiles.

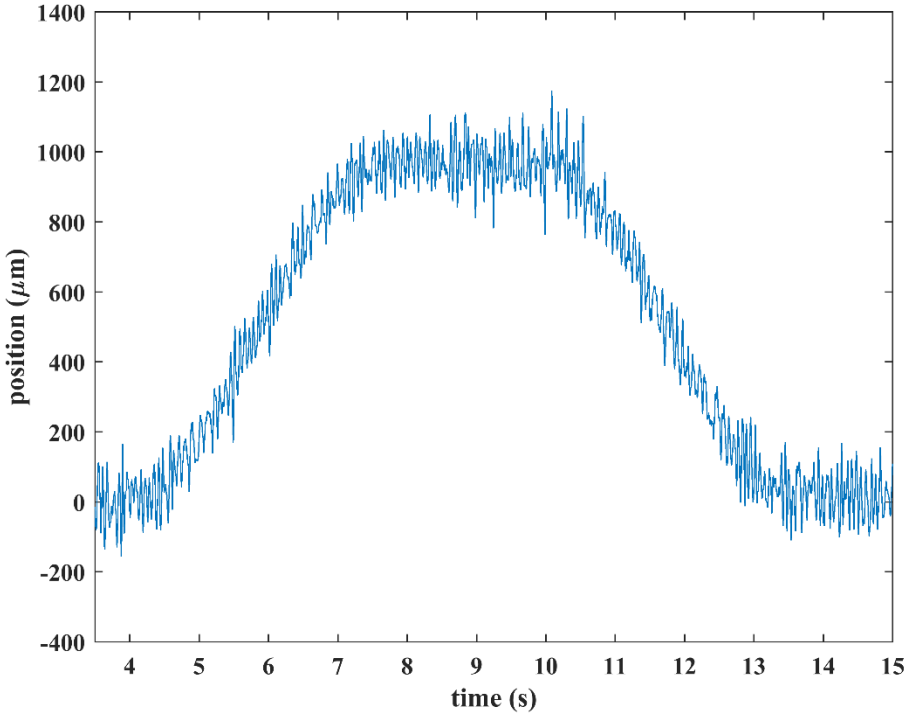



Figure 43: 1-mm continuous step-and-scan motion using red color strip

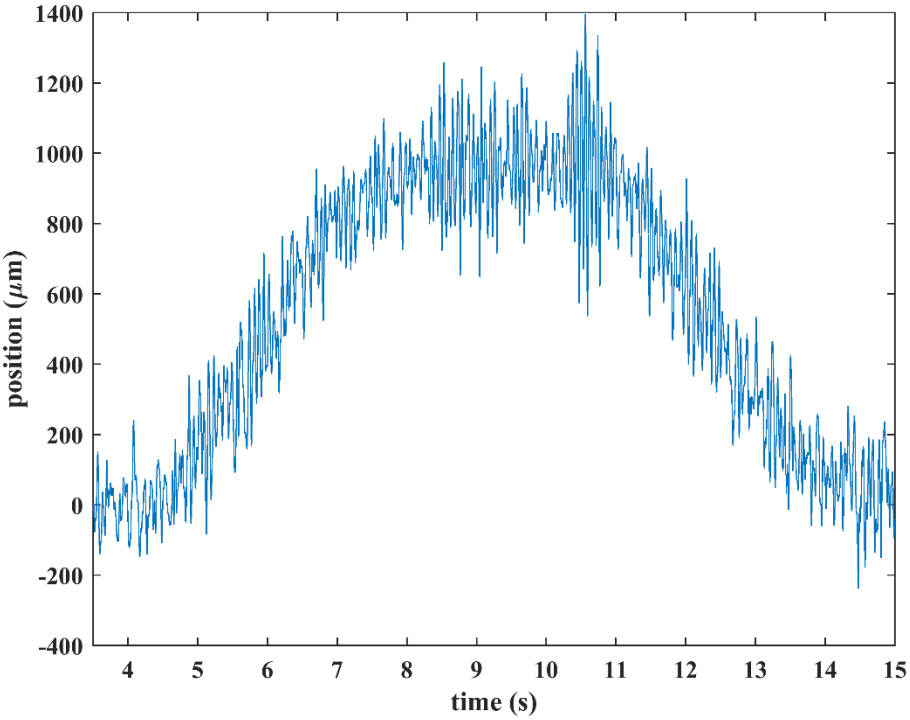



Figure 44: 1-mm continuous step-and-scan motion using blue color strip

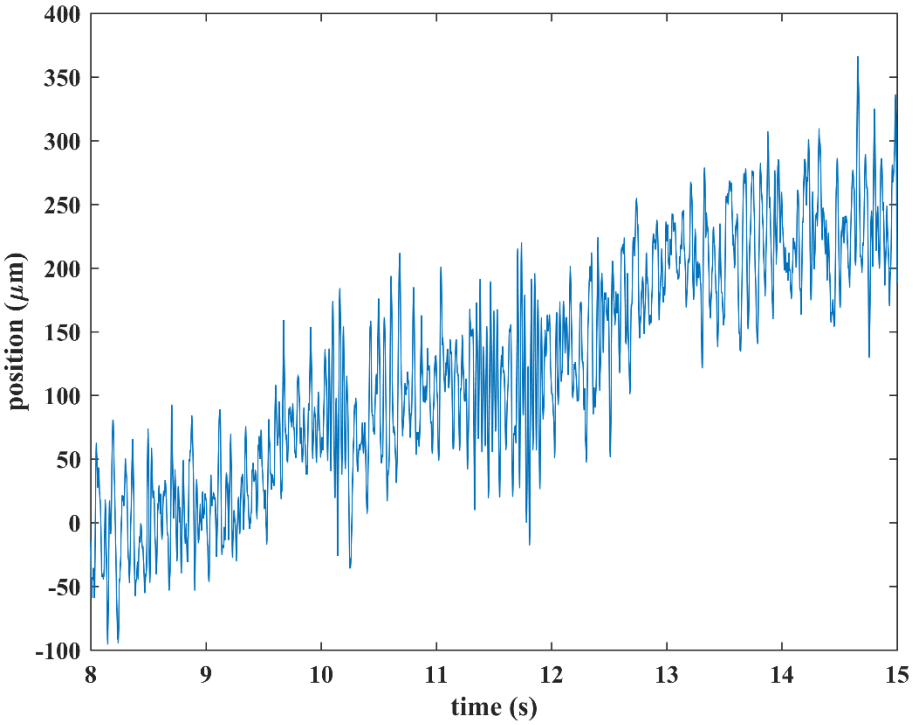



Figure 45: Consecutive 100-µm steps

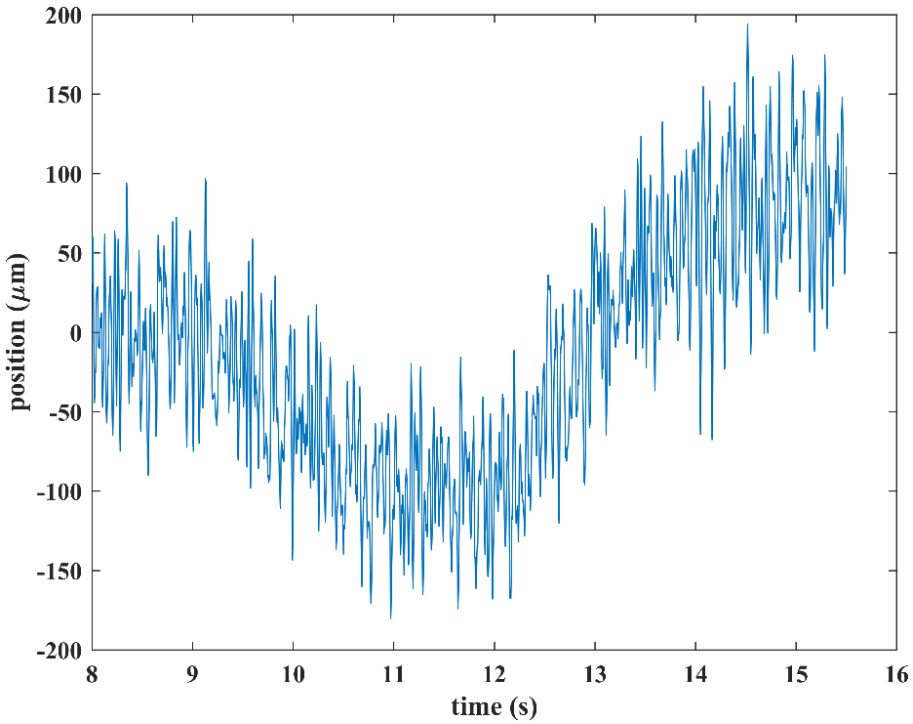



Figure 46: –100-µm step followed by 200-µm step

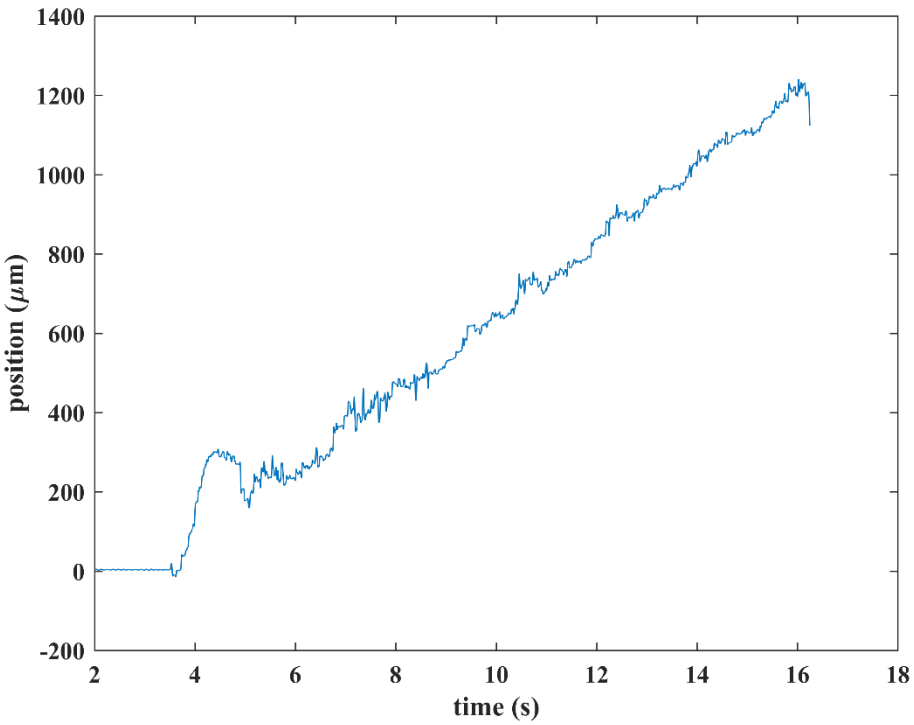



Figure 47: 1 mm continuous motion using step size of 25 nm

## 6.6. Maximum Speed Using RGB Sensor

In Figure 48 the position response of the platen completing a 5-mm step movement in the *y*-axis is shown. This response was used to determine the achieved speed of the system using the RGB sensor in the feedback loop. The maximum speed during the course of the run was found to be 0.01 m/s.

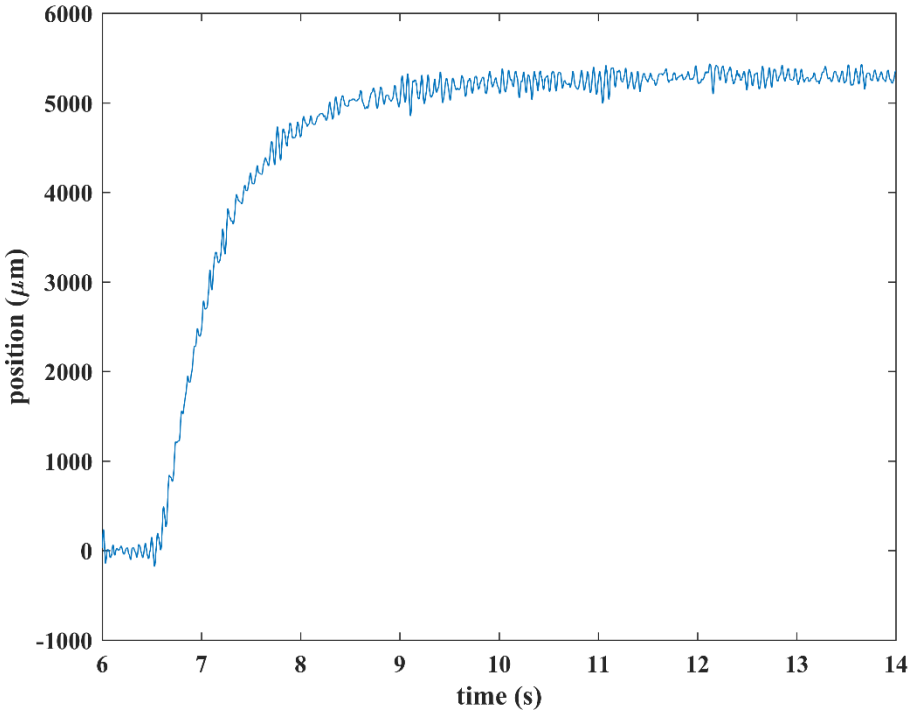



Figure 48: Position profile used to determine achieved maximum speed using RGB sensor

# CHAPTER VII

# CONCLUSIONS AND FUTURE WORK

## 7.1. Conclusions

In this thesis a position control algorithm for 1-D movement, using a red linearized color strip and RGB sensor, was proposed and implemented. The color strip, used to provide feedback to the RGB sensor, was linearized by applying the inverse method. Through a series of iterations, the initial nonlinearities between the RGB sensor and color strip were successfully minimized. The linearization process was repeated using blue as the color to be linearized.

A lead-PI controller was designed and implemented on a modified 6-DOF maglev test bed with the laser interferometers in the *y*-axis being replaced with the RGB sensor. The RGB sensor was mounted onto the moving platen using a cantilever arm and held a distance away from the platen with the sensor directly above a color strip with varying intensities of the color red. Both step responses and continuous trapezoidal responses were used to test the motion capabilities of the control system. The minimum position resolution of the controller was found to be 30 μm, with an average noise amplitude in the position response of 250 μm. The controller developed for this system was found to be best suited for the continuously updating reference position found in the continuous motion responses. Such a motion is common in the semiconductor and manufacturing industries where step-and-scan motions are essential for part creation.

## 7.2. Future Work

Based on the work, presented in this thesis, there are several possible avenues for future work to be done. The most attainable of which would be the development of a more robust controller with the capabilities of attenuating the position noise seen in this work. The development and implementation of this controller would improve the response of the system significantly. Furthermore, work could be done to improve the linearization process in order to develop highly linear color strips of significant length, i.e. larger than 10 cm.

# REFERENCES


[1] T. Ito and S. Okazaki, "Pushing the limits of lithography," *Nature*, vol. 406, no. 6799, pp. 1027–1031, Aug. 2000.

[2] M. Quirk and J. Serda, *Semiconductor Manufacturing Technology*. Upper Saddle River, NJ: Prentice-Hall, 2001.

[3] J. Mecomber, D. Hurd, and P. Limbach, "Enhanced machining of micron-scale features in microchip molding masters by CNC milling," *International Journal of Machine Tools & Manufacturing*, vol. 45, no. 12, pp. 1542–1550, Oct. 2005.

[4] Y. Liu and T. Higuchi, "Precision positioning device utilizing impact force of combined piezo-pneumatic actuator," *IEEE/ASME Transactions on Mechatronics*, vol. 6, no. 4, pp. 467–473, Dec. 2001.

[5] R. Cannon Jr. and E. Schmitz, "Initial experiments on the end-point control of a flexible one-link robot" *International Journal of Robotics Research*, vol. 3, no. 3, pp. 62−75, Sep. 1984.

[6] W. Gao, S. Dejima, H. Shimizu, S. Kiyono, and Y. Tomita, "A surface motor-driven planar motion stage integrated with an XY$\theta$Z surface encoder for precision positioning," *Precision Engineering*, vol. 28, no. 3, pp. 329–337, Jul. 2004.

[7] C. Ge, Y. Liao, Z. He, Z. Lou, Z. Huang, M. Wan, X. Hu, G. Fan, Z. Liang, "Closed loop high precision position control system with optical scale," in *Proceedings of the SPIE 6624, International Symposium on Photoelectric Detection and Imaging 2007: Optoelectronic System Design, Manufacturing, and Testing,* Beijing, 662418, Mar. 2008.

[8] V. Nguyen and W.-J. Kim, "A two-phase framework for linear permanent-magnet machines and multi-axis stages with magnetic levitation," in *Proceedings of the 7th ASME Dynamic Systems and Controls Conference*, Paper no. 5936, Oct. 2014.

[9] T. Gevers and A. Smeulders, "Color-based object recognition," *Journal of the Pattern Recognition Society*, vol. 32, no. 3, pp. 453−464, Mar. 1999.

[10] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "Object tracking with an adaptive color-based particle filter," in *Proceedings of the DAGM 2002: Pattern Recognition. Lecture Notes in Computer Science*, vol. 2449, pp. 353−360, Oct. 2002.

[11] A. Sivanantha Raja and K. Sankaranarayana, "Performance analysis of a colorimeter designed with RGB color sensor," in *Proceedings of the International Conference on Intelligent and Advanced Systems 2007,* Kuala Lumpur, Nov. 2007, pp. 305−310.

[12] M. Nkomo and M. Collier, "A color-Sorting SCARA robotic arm," in *Proceedings of the 2012 2nd International Conference on Consumer Electronics, Communications, and Networks (CECNet),* Yichang, pp. 763−768, Apr. 2012.

[13] M. Seelye, G. Sen Gubta, D. Bailey, and J. Seelye, "low cost colour sensors for monitoring plant growth in a laboratory," in *Proceedings of the 2011 IEEE International Instrumentation and Measurement Technology Conference*, Binjiang, pp. 1−6, May 2011.

[14] T. Tanaka and S. Haruyama, "New position detection method using image sensor and visible light LEDs," in *Proceedings of the 2009 Second International Conference on Machine Vision*, Dubai, pp. 150−153, Dec. 2009.

[15] Y. Kwon and W.-J. Kim, "Development of a new high-resolution angle-sensing mechanism using an RGB sensor" *IEEE/ASME Transactions on Mechatronics.*, vol. 19, no. 5, pp. 1707–1715, Oct. 2014.

[16] W.-J. Kim, "High-precision planar magnetic levitation," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, Jun. 1997.

[17] V. Nguyen, "Universal framework for linear motors and multi-axis stages with magnetic levitation," Ph.D. dissertation, Texas A&M University, College Station, TX, Dec. 2015.

[18] D. Trumper, M. Williams, and T. Nguyen, "Magnet arrays for synchronous machines," in *Proceedings of the 1993 IEEE Industry Applications Society 28th Annual Meeting*, vol. 1, pp. 9–18, Oct. 1993.

[19] E. Schubert, *Light-Emitting Diodes, 2nd Ed.* Cambridge, U.K.: Cambridge Univ. Press, 2006, ch. 4, 16, 24.

[20] T. Komine and M. Nakagawa, "Fundamental analysis for visible-light communication system using LED lights," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 101−107, Feb. 2004.

[21] R. Waynant and M. Ediger, *Electro-Optics Handbook, 2nd Ed.* New York, NY: McGraw-Hill, 2000, ch. 1.

[22] G. Franklin, J. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems, 6th Ed.* Reading, MA: Addison-Wesley, 1994, ch. 8.

# APPENDIX

## A.1. C Code implemented in Code Composer

```
#include "dsp.h"
#include "C:\tic3x4x\c3x4x\cgtools\include\math.h"

void c_int01()
{
    unsigned long D1reading;
    long ADreading;
    long LSreading;

    tr_low();
    D1reading=*(unsigned long int *)AD_FIFO_D1;

    if (index2 <= 130000){
           *(unsigned int *)(POS4+index2) = D1reading;

           sensor_d2 = (D1reading & 0xffff0000) >> 16;
           if (sensor_d2 > 32767){
                  sensor_d2 = sensor_d2 - 65536;
           }

           ADreading=*(unsigned long int *)AD_FIFO_A1;
           *(unsigned int *)(POS1+index2) = ADreading;

           sensor_a1 = (ADreading & 0x0000ffff);
           if (sensor_a1 > 32767){
                  sensor_a1 = sensor_a1 - 65536;
           }

           sensor_a2 = (ADreading & 0xffff0000) >> 16;
           if (sensor_a2 > 32767){
                  sensor_a2 = sensor_a2 - 65536;
           }

           ADreading=*(unsigned long int *)AD_FIFO_B1;
           *(unsigned int *)(POS2+index2) = ADreading;

           sensor_b1 = (ADreading & 0x0000ffff);
           if (sensor_b1 > 32767){
                  sensor_b1 = sensor_b1 - 65536;
           }

           sensor_b2 = (ADreading & 0xffff0000) >> 16;
           if (sensor_b2 > 32767){
                  sensor_b2 = sensor_b2 - 65536;
           }

           ADreading=*(unsigned long int *)AD_FIFO_C1;
           *(unsigned int *)(POS3+index2) = ADreading;

           sensor_c1 = (ADreading & 0x0000ffff);
           if (sensor_c1 > 32767){
                  sensor_c1 = sensor_c1 - 65536;
           }

           sensor_c2 = (ADreading & 0xffff0000) >> 16;
           if (sensor_c2 > 32767){
```

```
            sensor_c2 = sensor_c2 - 65536;
      }
}

      if ((sensor_a1 >= 6075) | (sensor_a1 <= -6075)) {y1s_temp = y1s_tempb;}
      if ((sensor_a1 >= -6074) & (sensor_a1 <= 6074)){
            y1s_temp = -asin(sensor_a1/6074.9)*0.00784951;
      }

      y1 = 0.9048*y1 + 0.0952*y1s_temp;
      y2s_temp = asin(sensor_a2/6074.9)*0.00940116;
      y2 = 0.9048*y2 + 0.0952*y2s_temp;
      y = (y1-y2)/2;

      x1s_temp = -asin(sensor_b1/6074.9)*0.008085;
      x1 = 0.9048*x1 + 0.0952*x1s_temp;
      x2s_temp = asin(sensor_b2/6074.9)*0.008085;
      x2 = 0.9048*x2 + 0.0952*x2s_temp;
      x = (x2-x1)/2;

      /*
      0.00784951 = 0.008085/1.03;
      0.00940116 = 0.008085/0.86;
      thetaz = (-y1-y2+0.0127)/0.1397;
   */

      thetaz = (x1 + x2 - 0.0127)/0.1397;
      trigox = gama1*(x2-0.0742);
      trigoy = gama1*(y1-0.08995);

/*   Uses the ADCs for the RGB sensor */

/*
z10 = (3000+sensor_d2*0.00152588)*0.000001;
z20 = (3000+sensor_c1*0.00152588)*0.000001;
z30 = (3000+sensor_c2*0.00152588)*0.000001;

z1 = 0.7788*z1 + 0.2212*z10b;
z2 = 0.7788*z2 + 0.2212*z20b;
z3 = 0.7788*z3 + 0.2212*z30b;

z = (z3+z2)/2;
thetax = (z3-z1)/0.108;
thetay = (z2-z1)/0.108;
*/

if (index2<29998){
      *(unsigned int *)DA_FIFO_A1=( (unsigned int)(84) << 16 ) & 0xffff0000 ;
      *(unsigned int *)DA_FIFO_A2=( (unsigned int)(114) << 16 ) & 0xffff0000 ;

      *(unsigned int *)DA_FIFO_B1=( (unsigned int)(50) << 16 ) & 0xffff0000 ;
      *(unsigned int *)DA_FIFO_B2=( (unsigned int)(171) << 16 ) & 0xffff0000 ;

      *(unsigned int *)DA_FIFO_C1=( (unsigned int)(66) << 16 ) & 0xffff0000 ;
      *(unsigned int *)DA_FIFO_C2=( (unsigned int)(-23) << 16 ) & 0xffff0000 ;


      *(unsigned int *)DA_FIFO_D1=( (unsigned int)(0) << 16 ) & 0xffff0000 ;
      *(unsigned int *)DA_FIFO_D2=( (unsigned int)(6) << 16 ) & 0xffff0000 ;
}

if((index2 >= 29998)&(index2 < 30000)){
      ux = 0;
```

```
        uxb = 0;
        uxbb = 0;
        uxi = 0;
        uxd = 0;

        uy = 0;
        uyb = 0;
        uybb = 0;
        uyi = 0;
        uyd = 0;

        uthetaz = 0;
        uthetazi = 0;
        uthetazd = 0;

        uthetax = 0;
        uthetaxi = 0;
        uthetaxd = 0;

        uthetay = 0;
        uthetayi = 0;
        uthetayd = 0;

        uthetaxdb = 0;
        uthetaydb = 0;

        uz = 0;
        uzi = 0;
        uzd = 0;
        uzdb = 0;
        uzb = 0;
        uzbb = 0;

        uz_added = 0;
        uthetax_added = 0;
        uthetay_added = 0;

        x1_posb = 0;
        x2_posb = 0;
        y_posb = 0;

        x1_laser = 0;
        x2_laser = 0;
        y_laser = 0;

        x_laserb = 0;
        x_laserbb = 0;

        y_laserb = 0;
        y_laserbb = 0;

        y_r = 0;
        y_rb = 0;
        y_rbb = 0;
        red_new = 0;
        red_b = 0;

        y_r_new = 0;
        y_r_newb = 0;
        y_r_newbb = 0;

        thetaz_laserb = 0;
}
```

```
   if((index2 >= 30000)&(index2 < 59000)){

          ex = x1r - x;
                 exb = x1r - xb;
                 exbb = x1r - xbb;

                 ey = y1r - y;
                 eyb = y1r - yb;
                 eybb = y1r - ybb;

                 ethetaz = thetaz1r - thetaz;
                 ethetazb = thetaz1r - thetazb;

                 ez = z1r - z;
                 ezb = z1r - zb;


/*
          ux = 1.983*uxb - 0.983*uxbb + 754.9*ex - 1508*exb + 753*exbb;
          uy = 1.983*uyb - 0.983*uybb + 754.9*ey - 1508*eyb + 753*eybb;

                 uthetazi = uthetazi + 0.001*ethetazb;
          uthetazd = 500*(ethetaz-ethetazb);
          uthetaz = uthetazi + 3*ethetaz + uthetazd;

          uxi = uxi + 0.2*(ex + exb);
          uxd = 1000*(ex - exb);
          ux = 500*ex + uxi + uxd;
                 uyi = uyi + 0.2*(ey + eyb);
          uyd = 1000*(ey - eyb);
          uy = 500*ey + uyi + uyd;

          uthetazi = uthetazi + 0.001*ethetazb;
          uthetazd = 1000*(ethetaz-ethetazb);
          uthetaz = 2*ethetaz + uthetazi + uthetazd;
   */

          ux = 1.9657*uxb - 0.9657*uxbb + 3774*ex - 7528.96*exb + 3755*exbb;
          uy = 1.9657*uyb - 0.9657*uybb  + 3774*ey - 7528.96*eyb + 3755*eybb;
          uthetazi = uthetazi + 0.0015*ethetazb;
          uthetaz = uthetazi + 6*ethetaz + 1200*(ethetaz-ethetazb);

          sx = sin(trigox);
          cx = cos(trigox);
          sy = sin(trigoy);
          cy = cos(trigoy);

          i01 = sx*ux + 0.5*cx*(uz+uz_added) - 2*r*cx*uthetax + r*sx*uthetaz;
          i02 = -cx*ux + 0.5*sx*(uz+uz_added) - 2*r*sx*uthetax - r*cx*uthetaz;
          i03 = cy*uy - 0.5*sy*(uz+uz_added) + 2*r*sy*uthetay + r*cy*uthetaz;
          i04 = sy*uy + 0.5*cy*(uz+uz_added) - 2*r*cy*uthetay + r*sy*uthetaz;

          i05 = cx*ux - 0.5*sx*(uz+uz_added) - 2*r*sx*uthetax - r*cx*uthetaz;

          i06 = -sx*ux - 0.5*cx*(uz+uz_added) - 2*r*cx*uthetax + r*sx*uthetaz;

          i07 = -sy*uy - 0.5*cy*(uz+uz_added) - 2*r*cy*uthetay + r*sy*uthetaz;
          i08 = -cy*uy + 0.5*sy*(uz+uz_added) + 2*r*sy*uthetay + r*cy*uthetaz;


          if (i01>2.318) {i01 = 2.318;}
          if (i01<-2.318) {i01 = -2.318;}
```

```
if (i02>2.318) {i02 = 2.318;}
if (i02<-2.318) {i02 = -2.318;}

if (i03>2.318) {i03 = 2.318;}
if (i03<-2.318) {i03 = -2.318;}

if (i04>2.318) {i04 = 2.318;}
if (i04<-2.318) {i04 = -2.318;}

if (i05>2.318) {i05 = 2.318;}
if (i05<-2.318) {i05 = -2.318;}

if (i06>2.318) {i06 = 2.318;}
if (i06<-2.318) {i06 = -2.318;}

if (i07>2.318) {i07 = 2.318;}
if (i07<-2.318) {i07 = -2.318;}

if (i08>2.318) {i08 = 2.318;}
if (i08<-2.318) {i08 = -2.318;}

i1 = 14130*i01 + 84.46;
i2 = 13490*i02 + 114.20;
i3 = 14100*i03 + 50.14;
i4 = 13450*i04 + 171.50;
i5 = 13820*i05 + 66.55;
i6 = 13410*i06 - 23.49;
i7 = 13495*i07;
i8 = 13480*i08 + 6.49;

uxbb = uxb;
uxb = ux;

uybb = uyb;
uyb = uy;

*(unsigned int *)DA_FIFO_A1=( (unsigned int)(i1) << 16 ) & 0xffff0000 ;
*(unsigned int *)DA_FIFO_A2=( (unsigned int)(i2) << 16 ) & 0xffff0000 ;

*(unsigned int *)DA_FIFO_B1=( (unsigned int)(i3) << 16 ) & 0xffff0000 ;
*(unsigned int *)DA_FIFO_B2=( (unsigned int)(i4) << 16 ) & 0xffff0000 ;

*(unsigned int *)DA_FIFO_C1=( (unsigned int)(i5) << 16 ) & 0xffff0000 ;
*(unsigned int *)DA_FIFO_C2=( (unsigned int)(i6) << 16 ) & 0xffff0000 ;

*(unsigned int *)DA_FIFO_D1=( (unsigned int)(i7) << 16 ) & 0xffff0000 ;
*(unsigned int *)DA_FIFO_D2=( (unsigned int)(i8) << 16 ) & 0xffff0000 ;

/*
*(unsigned int *)DA_FIFO_A1=( (unsigned int)(84) << 16 ) & 0xffff0000 ;
*(unsigned int *)DA_FIFO_A2=( (unsigned int)(114) << 16 ) & 0xffff0000 ;

*(unsigned int *)DA_FIFO_B1=( (unsigned int)(50) << 16 ) & 0xffff0000 ;
*(unsigned int *)DA_FIFO_B2=( (unsigned int)(171) << 16 ) & 0xffff0000 ;

*(unsigned int *)DA_FIFO_C1=( (unsigned int)(66) << 16 ) & 0xffff0000 ;
*(unsigned int *)DA_FIFO_C2=( (unsigned int)(-23) << 16 ) & 0xffff0000 ;

*(unsigned int *)DA_FIFO_D1=( (unsigned int)(0) << 16 ) & 0xffff0000 ;
*(unsigned int *)DA_FIFO_D2=( (unsigned int)(6) << 16 ) & 0xffff0000 ;

*/
```

```
    }

    if((index2 >= 59000)&(index2 < 60000)){

           /* F = 4 for plane mirror optics */
           /* where lamda is a laser wavelength (632.991 nm) */
           /* lamda/(F*2^22*100ns)= 3.77292037e-7, */
           /* METERS-BIT approx 0.625 nm */
                  /* write down y_pos, x1_pos, x2_pos to memory and read by gel*/


           if(index2 == 59000){
                  laser_setup();
           }

           if((index2 >= 59500)&(index2 < 60000)){
                  tr_low();
                  *(unsigned long int *)0xb0300003=0x0041;
                  raw_y_pos  = (*(long int *)0xb0300048 << 16) & 0xffff0000;
                  raw_x1_pos = (*(long int *)0xb0310048 << 16) & 0xffff0000;
                  raw_x2_pos = (*(long int *)0xb0320048 << 16) & 0xffff0000;

                  raw_y_vel  = (*(long int *)0xb030004e << 16) & 0xffff0000;
                         raw_x1_vel = (*(long int *)0xb031004e << 16) & 0xffff0000;
                         raw_x2_vel = (*(long int *)0xb032004e << 16) & 0xffff0000;

                  tr_high();
                  raw_y_pos  |= ((*(long int *)0xb0300048 >> 16) & 0x0000ffff);
                  raw_x1_pos |= ((*(long int *)0xb0310048 >> 16) & 0x0000ffff);
                  raw_x2_pos |= ((*(long int *)0xb0320048 >> 16) & 0x0000ffff);

                         raw_y_vel  |= ((*(long int *)0xb030004e >> 16) &
0x0000ffff);
                         raw_x1_vel |= ((*(long int *)0xb031004e >> 16) &
0x0000ffff);
                         raw_x2_vel |= ((*(long int *)0xb032004e >> 16) &
0x0000ffff);

                  *(unsigned int *)(POSY+index2) = raw_y_pos;
                  *(unsigned int *)(POSX1+index2) = raw_x1_pos;
                  *(unsigned int *)(POSX2+index2) = raw_x2_pos;

                  y_pos=raw_y_pos*6.1815119987e-10;
                  x1_pos=raw_x1_pos*6.1815119987e-10;
                  x2_pos=raw_x2_pos*6.1815119987e-10;

                  y_vel = raw_y_vel * 3.77292037e-7;
                  x1_vel= raw_x1_vel* 3.77292037e-7;
                         x2_vel= raw_x2_vel* 3.77292037e-7;

                         x1_laser = 0.7788*x1_laser + 0.2212*x1_posb;
                         x2_laser = 0.7788*x2_laser + 0.2212*x2_posb;
                         /*y_laser = 0.7788*y_laser + 0.2212*y_posb;*/
                  y_laser = 0;
                         x_laser = (x1_laser + x2_laser)/2;
                         thetaz_laser = (x1_laser - x2_laser)/0.1;

                  y_r_ini = sensor_c2;
           /*Capture initial red value of RGB sensor*/

           }

                  y_1_ini = y1;
```

```
y_2_ini = y2;
y_I = 0;
y_II = 0;
error = 0.0001;
y_Hall = 0;

y_1_vel= (y1 - y1b)*4000;
       y_2_vel= (y2 - y2b)*4000;
y_1_vel_fil = 0.0385;
       y_2_vel_fil = -0.006;

       flag = 210;

       /*
       ex = 0 - x_laser;
       exb = 0 - x_laserb;

       ey = 0 + y_laser;
       eyb = 0 + y_laserb;

       ethetaz = 0 - thetaz_laser;
       ethetazb = 0 - thetaz_laserb;
       */
x2_ini = x2;
y1_ini = y1;
       x2r = x1r;
y2r = y1r;
       thetaz2r = thetaz1r;
       thetax1r = thetax;
       thetay1r = thetay;
       z1r = z;
z2r = z1r - 0.000005;
uxi = 0;
uxd = 0;
uyi = 0;
uyd = 0;

ex = x1r - x;
       exb = x1r - xb;
       exbb = x1r - xbb;

       ey = y1r - y;
       eyb = y1r - yb;
       eybb = y1r - ybb;

       ethetaz = thetaz1r - thetaz;
       ethetazb = thetaz1r - thetazb;

ux = 1.9657*uxb - 0.9657*uxbb + 3774*ex - 7528.96*exb + 3755*exbb;
uy = 1.9657*uyb - 0.9657*uybb  + 3774*ey - 7528.96*eyb + 3755*eybb;
uthetazi = uthetazi + 0.0015*ethetazb;
uthetaz = uthetazi + 6*ethetaz + 1200*(ethetaz-ethetazb);

       ez = z1r - z;
       ezb = z1r - zb;

       ethetax = thetax1r - thetax;
       ethetaxb = thetax1r - thetaxb;

       ethetay = thetay1r - thetay;
       ethetayb = thetay1r - thetayb;

y_posb = y_pos;
```

```
x1_posb = x1_pos;
x2_posb = x2_pos;

x_laserbb = x_laserb;
x_laserb = x_laser;
y_laserbb = y_laserb;
y_laserb = y_laser;
thetaz_laserb = thetaz_laser;

y_rbb=y_rb;
y_rb = y_r;

uxbb = uxb;
uxb = ux;
uybb = uyb;
uyb = uy;

sx = sin(trigox);
cx = cos(trigox);
sy = sin(trigoy);
cy = cos(trigoy);

i01 = sx*ux + 0.5*cx*(uz+uz_added) - 2*r*cx*uthetax + r*sx*uthetaz;
i02 = -cx*ux + 0.5*sx*(uz+uz_added) - 2*r*sx*uthetax - r*cx*uthetaz;
i03 = cy*uy - 0.5*sy*(uz+uz_added) + 2*r*sy*uthetay + r*cy*uthetaz;
i04 = sy*uy + 0.5*cy*(uz+uz_added) - 2*r*cy*uthetay + r*sy*uthetaz;
i05 = cx*ux - 0.5*sx*(uz+uz_added) - 2*r*sx*uthetax - r*cx*uthetaz;

i06 = -sx*ux - 0.5*cx*(uz+uz_added) - 2*r*cx*uthetax + r*sx*uthetaz;
i07 = -sy*uy - 0.5*cy*(uz+uz_added) - 2*r*cy*uthetay + r*sy*uthetaz;
i08 = -cy*uy + 0.5*sy*(uz+uz_added) + 2*r*sy*uthetay + r*cy*uthetaz;

if (i01>2.318) {i01 = 2.318;}
if (i01<-2.318) {i01 = -2.318;}

if (i02>2.318) {i02 = 2.318;}
if (i02<-2.318) {i02 = -2.318;}

if (i03>2.318) {i03 = 2.318;}
if (i03<-2.318) {i03 = -2.318;}

if (i04>2.318) {i04 = 2.318;}
if (i04<-2.318) {i04 = -2.318;}

if (i05>2.318) {i05 = 2.318;}
if (i05<-2.318) {i05 = -2.318;}

if (i06>2.318) {i06 = 2.318;}
if (i06<-2.318) {i06 = -2.318;}

if (i07>2.318) {i07 = 2.318;}
if (i07<-2.318) {i07 = -2.318;}

if (i08>2.318) {i08 = 2.318;}
if (i08<-2.318) {i08 = -2.318;}

i1 = 14130*i01 + 84.46;
i2 = 13490*i02 + 114.20;
i3 = 14100*i03 + 50.14;
i4 = 13450*i04 + 171.50;
i5 = 13820*i05 + 66.55;
i6 = 13410*i06 - 23.49;
i7 = 13495*i07;
```

```
        i8 = 13480*i08 + 6.49;

        *(unsigned int *)DA_FIFO_A1=( (unsigned int)(i1) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_A2=( (unsigned int)(i2) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_B1=( (unsigned int)(i3) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_B2=( (unsigned int)(i4) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_C1=( (unsigned int)(i5) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_C2=( (unsigned int)(i6) << 16 ) & 0xffff0000 ;

        *(unsigned int *)DA_FIFO_D1=( (unsigned int)(i7) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_D2=( (unsigned int)(i8) << 16 ) & 0xffff0000 ;
        /*
        *(unsigned int *)DA_FIFO_A1=( (unsigned int)(84) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_A2=( (unsigned int)(114) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_B1=( (unsigned int)(50) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_B2=( (unsigned int)(171) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_C1=( (unsigned int)(66) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_C2=( (unsigned int)(-23) << 16 ) & 0xffff0000 ;

        *(unsigned int *)DA_FIFO_D1=( (unsigned int)(0) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_D2=( (unsigned int)(6) << 16 ) & 0xffff0000 ;

        */
}

/* ONLY CHANGE/MODIFY AFTER THIS*/
if((index2 >= 60000)&(index2 < 129999)){
        tr_low();
        *(unsigned long int *)0xb0300003=0x0041;
        raw_y_pos  = (*(long int *)0xb0300048 << 16) & 0xffff0000;
        raw_x1_pos = (*(long int *)0xb0310048 << 16) & 0xffff0000;
        raw_x2_pos = (*(long int *)0xb0320048 << 16) & 0xffff0000;
        raw_y_vel  = (*(long int *)0xb030004e << 16) & 0xffff0000;
                raw_x1_vel = (*(long int *)0xb031004e << 16) & 0xffff0000;
                raw_x2_vel = (*(long int *)0xb032004e << 16) & 0xffff0000;

        tr_high();
        raw_y_pos  |= ((*(long int *)0xb0300048 >> 16) & 0x0000ffff);
        raw_x1_pos |= ((*(long int *)0xb0310048 >> 16) & 0x0000ffff);
        raw_x2_pos |= ((*(long int *)0xb0320048 >> 16) & 0x0000ffff);
                raw_y_vel  |= ((*(long int *)0xb030004e >> 16) & 0x0000ffff);
                raw_x1_vel |= ((*(long int *)0xb031004e >> 16) & 0x0000ffff);
                raw_x2_vel |= ((*(long int *)0xb032004e >> 16) & 0x0000ffff);

        *(unsigned int *)(POSY+index2) = raw_y_pos;
        *(unsigned int *)(POSX1+index2) = raw_x1_pos;
        *(unsigned int *)(POSX2+index2) = raw_x2_pos;

        y_pos=raw_y_pos*6.1815119987e-10;
        x1_pos=raw_x1_pos*6.1815119987e-10;
        x2_pos=raw_x2_pos*6.1815119987e-10;
        y_vel = raw_y_vel * 3.77292037e-7;
        x1_vel= raw_x1_vel* 3.77292037e-7;

        x2_vel= raw_x2_vel* 3.77292037e-7;
                x1_laser = 0.7788*x1_laser + 0.2212*x1_posb;
                x2_laser = 0.7788*x2_laser + 0.2212*x2_posb;
                /*y_laser = 0.7788*y_laser + 0.2212*y_posb;*/
        y_laser = 0;
                x_laser = (x1_laser + x2_laser)/2;
                thetaz_laser = (x1_laser - x2_laser)/0.1;

                red = sensor_c2;  /*Convert sensor output from int to float*/
```

```
                y_r_ini_new = (y_r_ini);
          y_r_zero = 1642.6894;  /*Zero value of curve fit*/
          y_r_of = y_r_ini_new - y_r_zero;  /*Shift the zero of the curve fit to
                the initial value found earlier*/
          y_r = -0.00006812*(red - y_r_of) + 0.1119;  /*Calculate position*/

y_r_new = 0.94*y_r_new + 0.03*y_r + 0.03*y_rb;
/*Digital low-pass filter*/

          y_1_vel= (y1 - y1b)*4000;
                y_2_vel= (y2 - y2b)*4000;
          y_1_vel_fil = 0.9753*y_1_vel_fil + 0.0247*y_1_vel;
                y_2_vel_fil = 0.9753*y_2_vel_fil + 0.0247*y_2_vel;

          if(index2 >= 70000){
                        if((y1-y2)*(y1-y2) < 0.0000000025){
                        if ((y_1_vel_fil > 0.001) & (y_2_vel_fil < -0.001)){
                              if (y1 > 0){
                                          flag = 210;
/* change to y2 active, go forward, minus*/
                              }
                              if (y1 < 0){
                                          flag = 101;
                                    }
                              }
                        if ((y_1_vel_fil < -0.001) & (y_2_vel_fil > 0.001)){
                                    if (y1 < 0){
                                    flag = 211;
                                    }
                                    if (y1 > 0){
                                    flag = 100;
                                    /* change to y1 active, go backward, minus*/
                                    }
                        }
                        }
                        if((y1+y2)*(y1+y2) < 0.0000000025){
                        if ((y_1_vel_fil < -0.001) & (y_2_vel_fil < -0.001)){
                                    if (y1 > 0){
                                    flag = 110;
                                     /* change to y1 active, go forward, minus*/
                                    }
                                    if (y1 < 0){
                                    flag = 200;
                                    }
                        }
                        if ((y_1_vel_fil > 0.001) & (y_2_vel_fil > 0.001)){
                                    if (y1 > 0){
                                    flag = 201;
                                    }
                                    if (y1 < 0){
                                    flag = 111;
                                    /* change to y1 active, go forward, plus*/
                                    }
                        }
                        }
                        if ((flag == 111) | (flag == 100)){     /* y1 active*/
                        y_Hall = y_II + (y1 - y_1_ini);
                        y_2_ini = y2;
                        y_I = y_Hall;
                        }
                        if ((flag == 210) | (flag == 201)){     /* y2 active*/
                        y_Hall = y_I - (y2 - y_2_ini);
```

```
                y_1_ini = y1;
                y_II = y_Hall;

                }
                if ((flag == 110) | (flag == 101)){     /* y1 active*/
                y_Hall = y_II - (y1 - y_1_ini);
                y_2_ini = y2;
                y_I = y_Hall;
                }
                if ((flag == 211) | (flag == 200)){     /* y2 active*/
                y_Hall = y_I + (y2 - y_2_ini);
                y_1_ini = y1;
                y_II = y_Hall;
                }
}
y_r_out = y_r_new*100000;
*(unsigned int *)(POS_U + index2) = y_r_out;

ex = x2r - x;
       exb = x2r - xb;
exbb = x2r - xbb;
       ey = y2r - y;
       eyb = y2r - yb;
       eybb = y2r - ybb;
       ethetaz = thetaz2r - thetaz;
       ethetazb = thetaz2r - thetazb;
       ethetax = thetax1r - thetax;
       ethetaxb = thetax1r - thetaxb;
       ethetay = thetay1r - thetay;
       ethetayb = thetay1r - thetayb;
ez = z2r - z;
ezb = z2r - zb;

if((index2 >= 81000) & (index2 < 101000)){            /*Desired motion*/
       xref = 0;
       yref = 0;
       /*yref = yref + 0.0000001;*/
       thetazref = 0;
}

if((index2 >= 101000) & (index2 < 125000)){
       xref = 0;
       yref = 0.0001;
       /*yref = yref + 0.0000001;*/
       thetazref = 0;
}

/*
if((index2 >= 106000) & (index2 < 116000)){
       xref = 0;
       /*yref = .001;*
       yref = yref - 0.0000001;
       thetazref = 0;
}


if((index2 >= 116000) & (index2 < 125000)){
       xref = 0;
       yref = 0;
       /*yref = yref - 0.0000001;*
       thetazref = 0;
}
*/
```

```
          ex_laser = xref - x_laser;
          ex_laserb = xref - x_laserb;
          ex_laserbb = xref - x_laserbb;

                ey_laser = yref - y_r_new;
          /* Designate which sensor controls feedback*/
                ey_laserb = yref - y_r_newb;
                ey_laserbb = yref - y_r_newbb;
          ethetaz_laser = thetazref - thetaz_laser;
          ethetaz_laserb = thetazref - thetaz_laserb;
          ey_laser2 = yref + y_laser;
                ey_laserb2 = yref + y_laserb;
                ey_laserbb2 = yref + y_laserbb;

          /*
          ey_laser2 = yref + y_laser;
                ey_laserb2 = yref + y_laserb;
                ey_laserbb2 = yref + y_laserbb;
                ey_laser = yref - y_r_new;
                ey_laserb = yref - y_r_newb;
                ey_laserbb = yref - y_r_newbb;

          */

          /*Controllers*/
          if(index2 < 71000){
                ux = 1.9657*uxb - 0.9657*uxbb + 3774*ex - 7528.96*exb + 3755*exbb;
                /*7528.96*/
                      uy = 1.9657*uyb - 0.9657*uybb + 3774*ey - 7528.98*eyb +
3755*eybb;
                uthetazi = uthetazi + 0.0017*ethetazb;
                uthetaz = uthetazi + 7*ethetaz + 1275*(ethetaz-ethetazb);
                sx = sin(trigox);
                cx = cos(trigox);
                sy = sin(trigoy);
                cy = cos(trigoy);
                x2_ini = x2;
                y1_ini = y1;
                }
          if(index2 >= 71000){
ux = 1.9657*uxb - 0.9657*uxbb + 3774*ex_laser - 7528.96*ex_laserb +
3755*ex_laserbb;
uy = 1.9279*uyb - 0.9279*uybb  + 10742*ey_laser - 21440.975*ey_laserb +
10699*ey_laserbb;
/*This is where RGB sensor takes over*/

                uthetazi = uthetazi + 2*ethetaz_laserb;
                uthetazd = -12*(x1_vel-x2_vel);
                uthetaz = uthetazi + 100*ethetaz_laser + uthetazd;

                trigox_laser = gama1*(x2_ini - 0.0742 + x_laser);
                trigoy_laser = gama1*(y1_ini - 0.08995 - y_r_new);
                sx = sin(trigox_laser);
                cx = cos(trigox_laser);
                sy = sin(trigoy_laser);
                cy = cos(trigoy_laser);
          }

          if(index2 >= 79000){
ux = 1.9657*uxb - 0.9657*uxbb + 3774*ex_laser - 7528.96*ex_laserb +
3755*ex_laserbb;
```

```
uy = 1.9279*uyb - 0.9279*uybb  + 10742*ey_laser - 21440.985*ey_laserb +
10699*ey_laserbb;   /*RGB sensor*/

                        uthetazi = uthetazi + 2*ethetaz_laserb;
                  uthetazd = -12*(x1_vel-x2_vel);
                  uthetaz = uthetazi + 100*ethetaz_laser + uthetazd;

                        trigox_laser = gama1*(x2_ini - 0.0742 + x_laser);
                  trigoy_laser = gama1*(y1_ini - 0.08995 - y_r_new);
                  sx = sin(trigox_laser);
                  cx = cos(trigox_laser);
                  sy = sin(trigoy_laser);
                  cy = cos(trigoy_laser);
            }

i01 = sx*ux + 0.5*cx*(uz+uz_added) - 2*r*cx*(uthetax+uthetax_added) + r*sx*uthetaz;
i02 = -cx*ux + 0.5*sx*(uz+uz_added) - 2*r*sx*(uthetax+uthetax_added) -
r*cx*uthetaz;
i03 = cy*uy - 0.5*sy*(uz+uz_added) + 2*r*sy*(uthetay+uthetay_added) + r*cy*uthetaz;
i04 = sy*uy + 0.5*cy*(uz+uz_added) - 2*r*cy*(uthetay+uthetay_added) + r*sy*uthetaz;
i05 = cx*ux - 0.5*sx*(uz+uz_added) - 2*r*sx*(uthetax+uthetax_added) - r*cx*uthetaz;

i06 = -sx*ux - 0.5*cx*(uz+uz_added) - 2*r*cx*(uthetax+uthetax_added) +
r*sx*uthetaz;
i07 = -sy*uy - 0.5*cy*(uz+uz_added) - 2*r*cy*(uthetay+uthetay_added) +
r*sy*uthetaz;
i08 = -cy*uy + 0.5*sy*(uz+uz_added) + 2*r*sy*(uthetay+uthetay_added) +
r*cy*uthetaz;

            /*
            *(unsigned int *)(POS_U + index2 - 60000) = raw_x1_vel;
            utemp = 10000*y_Hall;
            *(unsigned int *)(POS_U + index2 - 60000) = utemp;
            */

            if (i01>2.318) {i01 = 2.318;}
            if (i01<-2.318) {i01 = -2.318;}

            if (i02>2.318) {i02 = 2.318;}
            if (i02<-2.318) {i02 = -2.318;}

            if (i03>2.318) {i03 = 2.318;}
            if (i03<-2.318) {i03 = -2.318;}

            if (i04>2.318) {i04 = 2.318;}
            if (i04<-2.318) {i04 = -2.318;}

            if (i05>2.318) {i05 = 2.318;}
            if (i05<-2.318) {i05 = -2.318;}

            if (i06>2.318) {i06 = 2.318;}
            if (i06<-2.318) {i06 = -2.318;}

            if (i07>2.318) {i07 = 2.318;}
            if (i07<-2.318) {i07 = -2.318;}

            if (i08>2.318) {i08 = 2.318;}
            if (i08<-2.318) {i08 = -2.318;}

            i1 = 14130*i01 + 84.46;
                  i2 = 13490*i02 + 114.20;
            i3 = 14100*i03 + 50.14;
            i4 = 13450*i04 + 171.50;
```

```
        i5 = 13820*i05 + 66.55;
        i6 = 13410*i06 - 23.49;
        i7 = 13495*i07;
        i8 = 13480*i08 + 6.49;
        y_posb = y_pos;
        x1_posb = x1_pos;
        x2_posb = x2_pos;

        x_laserbb = x_laserb;
        x_laserb = x_laser;
        y_laserbb = y_laserb;
        y_laserb = y_laser;
        thetaz_laserb = thetaz_laser;


        y_r_newbb = y_r_newb;
        y_r_newb = y_r_new;
        y_rbb = y_rb;
        y_rb = y_r;
        red_b = red;

        y_Hallb = y_Hall;

        uxbb = uxb;
        uxb = ux;
        uybb = uyb;
        uyb = uy;

        *(unsigned int *)DA_FIFO_A1=( (unsigned int)(i1) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_A2=( (unsigned int)(i2) << 16 ) & 0xffff0000 ;

        *(unsigned int *)DA_FIFO_B1=( (unsigned int)(i3) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_B2=( (unsigned int)(i4) << 16 ) & 0xffff0000 ;

        *(unsigned int *)DA_FIFO_C1=( (unsigned int)(i5) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_C2=( (unsigned int)(i6) << 16 ) & 0xffff0000 ;

        *(unsigned int *)DA_FIFO_D1=( (unsigned int)(i7) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_D2=( (unsigned int)(i8) << 16 ) & 0xffff0000 ;
        /*
        *(unsigned int *)DA_FIFO_A1=( (unsigned int)(84) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_A2=( (unsigned int)(114) << 16 ) & 0xffff0000 ;

        *(unsigned int *)DA_FIFO_B1=( (unsigned int)(50) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_B2=( (unsigned int)(171) << 16 ) & 0xffff0000 ;

        *(unsigned int *)DA_FIFO_C1=( (unsigned int)(66) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_C2=( (unsigned int)(-23) << 16 ) & 0xffff0000 ;

        *(unsigned int *)DA_FIFO_D1=( (unsigned int)(0) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_D2=( (unsigned int)(6) << 16 ) & 0xffff0000 ;
        */
}

if((index2 >= 129999)&(index2 < 130000)){

        *(unsigned int *)DA_FIFO_A1=( (unsigned int)(84) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_A2=( (unsigned int)(114) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_B1=( (unsigned int)(50) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_B2=( (unsigned int)(171) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_C1=( (unsigned int)(66) << 16 ) & 0xffff0000 ;
        *(unsigned int *)DA_FIFO_C2=( (unsigned int)(-23) << 16 ) & 0xffff0000 ;
```

```
            *(unsigned int *)DA_FIFO_D1=( (unsigned int)(0) << 16 ) & 0xffff0000 ;
            *(unsigned int *)DA_FIFO_D2=( (unsigned int)(6) << 16 ) & 0xffff0000 ;
    }

    y1s_tempb = y1s_temp;
    y2s_tempb = y2s_temp;

    x1s_tempb = x1s_temp;
    x2s_tempb = x2s_temp;

    z10b = z10;
    z20b = z20;
    z30b = z30;

    xbb = xb;
    xb = x;

    ybb = yb;
    yb = y;
    y1b = y1;
    y2b = y2;

    thetazb = thetaz;
    zb = z;
    thetaxb = thetax;
    thetayb = thetay;
    index2 = index2 + 1;
    MX_Int_Clr= 0x20000029;
    *(unsigned int *)MX_Int_Clr=0x0;
}
```

## A.2. Linearization Code

```
clear, clc

path(path,'f:\Mechatronics Lab\Matlab')

stc = textread('cnew3232.txt','%s');
std = textread('dnew3232.txt','%s');


file1 = '3232.xlsx';
full_intensity_values = 'full_strip_RGB_intensity_02.xlsx';
fiv = xlsread(full_intensity_values,'D3:D403');
fiv = fiv.';
position = xlsread(file1,'C1:C32001');    %These values need to be updated
with position and sensor output
rsensor = xlsread(file1,'B1:B32001');


length_strip = 0.04;  % The length of the original color strip
initial_pos = -.01;
num_of_rows = 50;
sections = 114;  %Number of different RGB sections in printout
total_sections = 401;
initial_section = 171;
final_section = 285;
red_v2 = '-1_3cm.xlsx';  % The file used to create the original color
strip
R_v2 = zeros(num_of_rows,sections,3);
R_v2(:,:,1) = xlsread(red_v2)/500;
% figure(1)
% image(R_v2);
step_length = (length_strip/sections);

R_strip = zeros(2,sections);
position_strip = zeros(1,sections);

for i = 1:(sections-1)

    position_strip(1) = initial_pos;
    position_strip(i+1) = position_strip(i)+step_length;

end

for i = 1:sections

    R_strip(1,i) = R_v2(1,i,1);
    R_strip(2,i) = position_strip(i);

end
```

```
% position = xlsread(file1,'C1:C49001');    %These values need to be
updated with position and sensor output
% rsensor = xlsread(file1,'B1:B49001');

numbz1 = 1:1:65000;
numbz2 = 1:1:65000;
numbz3 = 1:1:65000;


t1 = 1:1:65000;
t11 = (t1-1)/4000;

for i = 1:65000

     tempc = char(stc(i+5));
     tempd = char(std(i+5));



     numbz3(i) = hex2dec(strcat(tempc(3),tempc(4),tempc(5),tempc(6)));
%channel C2
     numbz2(i) = hex2dec(strcat(tempc(7),tempc(8),tempc(9),tempc(10)));
%channel C1
     numbz1(i) = hex2dec(strcat(tempd(3),tempd(4),tempd(5),tempd(6)));
%channel D2

end

for i = 1:65000
    if numbz1(i) > 2^15
        numbz1(i) = numbz1(i)-2^16;
    end
    if numbz2(i) > 2^15
        numbz2(i) = numbz2(i)-2^16;
    end
    if numbz3(i) > 2^15
        numbz3(i) = numbz3(i)-2^16;
    end

%
end

t2 = 1:1:65000;
t22 = (t2-1)/4000;
red = numbz3.';
t = t22.';


%%
L = length_strip;   %m
n = sections;
dis_step = L/n;
discrete_distance = floor(length(rsensor)/n);    %Changed because of error
from rsensor to length of rsensor manually inputed
```

```
discrete_position = zeros(1,n);
discrete_sensor = zeros(1,n);
discrete_position(1) = position(1);
discrete_sensor(1) = rsensor(1);
for i = 1:(n-1)

    discrete_position(i+1) = position(i*discrete_distance);
    discrete_sensor(i+1) = rsensor(i*discrete_distance);

end

linearized_position = zeros(1,n);
linearized_sensor = zeros(1,n);

linearized_sensor(1) = max(discrete_sensor);
N = find(discrete_sensor==linearized_sensor(1));
linearized_position(1) = discrete_position(N);
linearized_sensor(n) = min(discrete_sensor);
M = find(discrete_sensor==linearized_sensor(1));
linearized_position(1) = discrete_position(M);

slope_key = (linearized_sensor(1) - linearized_sensor(n))/L;
slope = zeros(1,n-2);

for i = 2:(n-1)

    for j=1:(n-2)

        slope(j) = ((linearized_sensor(1) - discrete_sensor(j+1))/((i-
1)*dis_step));

        if slope(j) < 0
            slope(j) = abs(slope(j));
        end

    end



    slope_ini = abs(slope(1));
    for k = 2:(n-2)
    if abs(slope_key - slope(k)) <= abs(slope_key - slope_ini)
        slope_ini = slope(k);
    end
    end
    T = find(slope==slope_ini);
    R = T(1);
    linearized_sensor(i) = discrete_sensor(R+1);
    linearized_position(i) = discrete_position(R+1);
    linearized_position_actual = (0:dis_step:(L-dis_step));

end
% plot(linearized_position_actual,linearized_sensor)
```

```
position_linear = zeros(1,n);
for j = 1:n

    T = find(rsensor==linearized_sensor(j));
    R = T(1);
    position_linear(j) = position(R);

end
for j = 1:n
    for k = 1:n

   A(k) = abs(position_linear(j) - position_strip(k));

    end

   B = A.';
   [M,I] = min(B);
   point = I;

   linearized_pos(j) = position_strip(point);

end

for j = 1:n

    T = find(R_strip(2,:)==linearized_pos(j));
    R = T(1);
    sensor_RGB(j) = R_strip(1,R);

end

for i = 1:length(sensor_RGB)

    fiv(i+(initial_section-1)) = sensor_RGB(i);

end

R_final = zeros(num_of_rows,total_sections,3);

for i = 1:total_sections

    R_final(1,i,1) = fiv(i);

    for j = 2:num_of_rows

       R_final(j,i,1) = R_final(1,i,1);

    end

end
figure(42)
image(R_final);
```

```
R_s = zeros(num_of_rows,n,3);

for i = 1:n

    R_s(1,i,1) = sensor_RGB(i);

    for j = 2:num_of_rows

       R_s(j,i,1) = R_s(1,i,1);

    end

end
figure(43)
image(R_s);

%%
file2 = '3286_linearize_red.xlsx';
opt_linearized = xlsread(file2,'B1:B401');
opt_linearized = opt_linearized.';

R_linearized = zeros(num_of_rows,total_sections,3);

for i = 1:total_sections

    R_linearized(1,i,1) = opt_linearized(i);

    for j = 2:num_of_rows

       R_linearized(j,i,1) = R_linearized(1,i,1);

    end

end
figure(44)
image(R_linearized);

%%
red='Calibration_Red_2.xlsx';
red_v2 = 'Red_linear_range.xlsx';
% R=zeros(25,401,3);
% R(:,:,1)=xlsread(red)/500;
% figure(1)
% image(R)
%
% blue='Calibration_Blue.xlsx';
%
% B=zeros(25,401,3);
% B(:,:,3)=xlsread(blue)/500;
% figure(2)
% image(B)

R_v2 = zeros(25,401,3);
R_v2(:,:,1) = xlsread(red)/500;
```

```
for i =1:1:50
    R_v2(:,i,1) = 199 + i;
%     R_v2(i,:,1) = R_v2(i,1,1);
end
R_v2(:,:,1) = xlsread(red_v2)/500;
R_v2(:,:,1) = R_v2(:,:,1)/500;
figure(3)
image(R_v2)
```