# OPTICAL DISPLACEMENT MEASUREMENT BY IMAGE CORRELATION

by

Nathan Lambert

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Mechanical Engineering

Charlotte

2019

Approved by:

______________________________

Dr. Steve Patterson

______________________________

Dr. Kevin Lawton

______________________________

Dr. Tom Suleski

# ABSTRACT

NATHAN LAMBERT. optical displacement measurement by image correlation.
(Under the direction of DR. STEVE PATTERSON)

This thesis describes the design and verification of a scanning, optical, displacement measurement instrument, which utilizes digital image correlation (DIC) to measure displacement. This system is intended to provide proof of concept for a technique to be incorportated into a optical creep measurement instrument. This system captures magnified images of a wire sample, and compares these images against a set of target images using normalized cross-correlation in order to locate a unique set of features on a wire sample. A linear encoder measures the displacement of the wire as it is translated axially in relation to the imaging system. The total distance between two points on the wire is computed by summing the global coordinate measurement from the linear encoder and the local coordinate scale measurement using image correlation. The components which were designed for this system include, a co-axially illuminated imaging system, flexure based mounting system which allows for high precision component alignment, and a high accuracy linear positioning system. The goal for this system is the capability of resolving the position of specific features on a wire, to 10 nm without the need for special preparation of the samples. The proof of concept prototype, performed a series of 40 DIC based position measurements, at a rate of one measurement per second, resulting in a repeatability uncertainty of 4.58 nm. This level of performance demonstrates the viability of this method of displacement measurement, for implementation into a complete creep measurement instrument. The primary sources of measurement uncertainty are thermal in nature and could be reduced with tighter environmental temperature control and further design for thermal stability.

# DEDICATION

I want to dedicate this thesis to my wife, Brittney. Her support during this process has been essential. Without it, this thesis would not be possible.

# ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Steven R. Patterson, whose incredible knowledge, guidance, and mentorship made this thesis possible. I would also like to thank Dr. Kevin M. Lawton, whose invaluable insight has been essential in the creation of this thesis. Additionally, Dr. Alan R. Freitag and the Graduate School at UNC Charlotte provided an immensely generous veteran graduate student assistantship. Micheal J. Coniglio of Heidenhain Corporation provided assistance and material support that is greatly appreciated.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1: INTRODUCTION

## 1.1 Purpose

Creep generally occurs over an extended period of time, and for this reason measuring creep can be a very slow process. To conduct repeated measurements with an instrument which only measures one sample at a time, might take several months or even years. Changes in laboratory environmental conditions can make comparing sequential test runs with such long time intervals between them difficult. Therefore, it is desirable to run several tests simultaneously. One method for achieving this would be to use multiple instruments to complete these tests; however, this can be very costly and variations from one instrument to another may also make comparing measurements challenging.

Another approach to addressing these issues is to develop a single instrument which can measure multiple wires during the same test. The aim of this project is to investigate the potential sources of uncertainty which arise in such an instrument. Figure (1.1) below shows a CAD model of the complete instrument assembly.

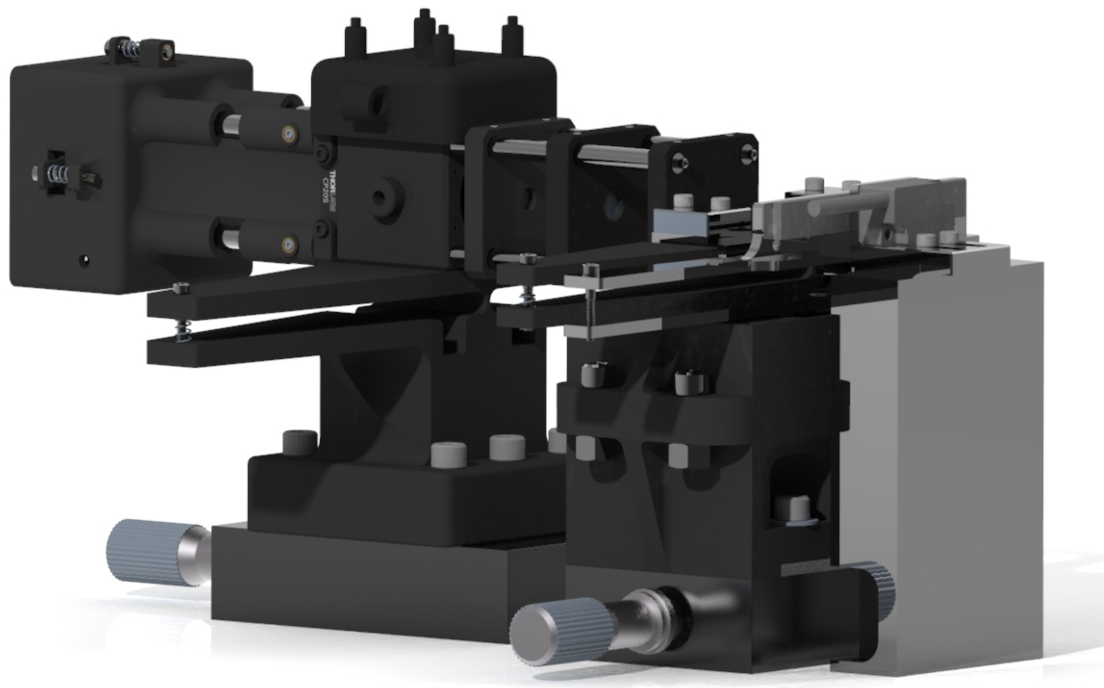


Figure 1.1: Complete displacement measurement instrument

## 1.2 Previous Work

One of the common methods for optical creep measurement currently in use involves taking a magnified image of an object and splitting the image into a large number of sub-images. The position of each sub-image is tracked using Digital Image Correlation (DIC). The distances between the sub-images are computed and used to calculate the total strain. Many techniques involve the application of a known pattern to the surface of the sample part, such as a pattern of dots, and subsequently imaging the sample (sometimes referred to as speckles, when applied randomly). One such pattern generation technique is described by Di Gioacchino *et al* [1]. They describe a technique for vapor deposition of nanometer scale gold particles to be used as markers for scanning electron microscopy. One technique described by Telfer, *et al.* [2] involves imaging the pattern from multiple angles and then employing photogrammetry techniques to develop a 3-dimensional model of the test object. The distances between the markings are measured using digital image correlation and a 3-dimensional strain field is then calculated. These techniques allows the pixel level tracking of material displacement. Sub-pixel algorithms are commonly employed, which allow for even greater resolution. One such method for achieving sub-pixel resolution is outlined by Sousa, *et al.* [3] The proposed method involves dividing sub-images into four smaller sub-images and then estimating the displacement of each sub-image using an optical flow based technique. In a paper by Debella-Gilo, *et al.* [4] several methods for attaining sub-pixel resolution while using DIC methods to track glacier flows with satellite imagery are explored. The methods investigated include interpolation methods using parabolic and gaussian models, as well as bicubic interpolation. Another method of creep measurement described by Harding, *et al.* [5] utilizes Moire fringes to generate a displacement signal as a sample creeps. Another novel approach to measuring creep in 304 stainless steel is described in an article by Elhoucine and Nagy [6]. This article outlines a method for creep measurement which utilizes "directional potential drop" resulting from the change in a sample's metallurgical structure during creep deformation.

### 1.2.1 Previous measurements at UNC Charlotte

The previous instrument configuration utilized a capacitance-based displacement sensor to measure the creep of a single wire with a nominal diameter of 0.003in. One of the drawbacks of this system is that the capacitance-based mechanism has limited dynamic range. The primary and tertiary creep phases experience large length changes, while the secondary phase changes at a much slower rate [7]. While the previous instrument can achieve the required nanometer level resolution, its total range is not sufficient to capture the full extent of the creep behavior. Another significant limitation of the previous system is the time it takes to run multiple tests because it measures only one wire at a time [8].

### 1.2.2 Instrument goal

The goal for this instrument is the ability to make optical displacement measurements of feature sets on a wire, with a resolution of 10 nm and a total range of no less than 1 mm. This instrument is a proof of concept, with the aim of gaining a better understanding of the likely uncertainties associated with measurements taken with an instrument of this type. Verifiying that this technique can produce the required performance for an optical displacement measurement instrument, demonstrates the value of this technique for integration into an optical creep measurement instrument, capable of measuring multiple wires simultaneously.

### 1.2.3 Overall instrument design

In order to achieve the desired dynamic range, a camera is mounted on a translation stage, facing a wire sample. The image from the camera is processed using DIC in order to measure the location of a unique set of surface features on the unmodified wire. Once these features are located, the wire is translated to its opposite end, and the camera finds the location of another set of unique features. The translation of the stage is measured by a 10 nm resolution linear encoder. The camera system cycles between the two sets of features on the wire and utilizes both the camera position measurement and the linear encoder measurement to calculate the total displacement between the two target feature sets on the wire.

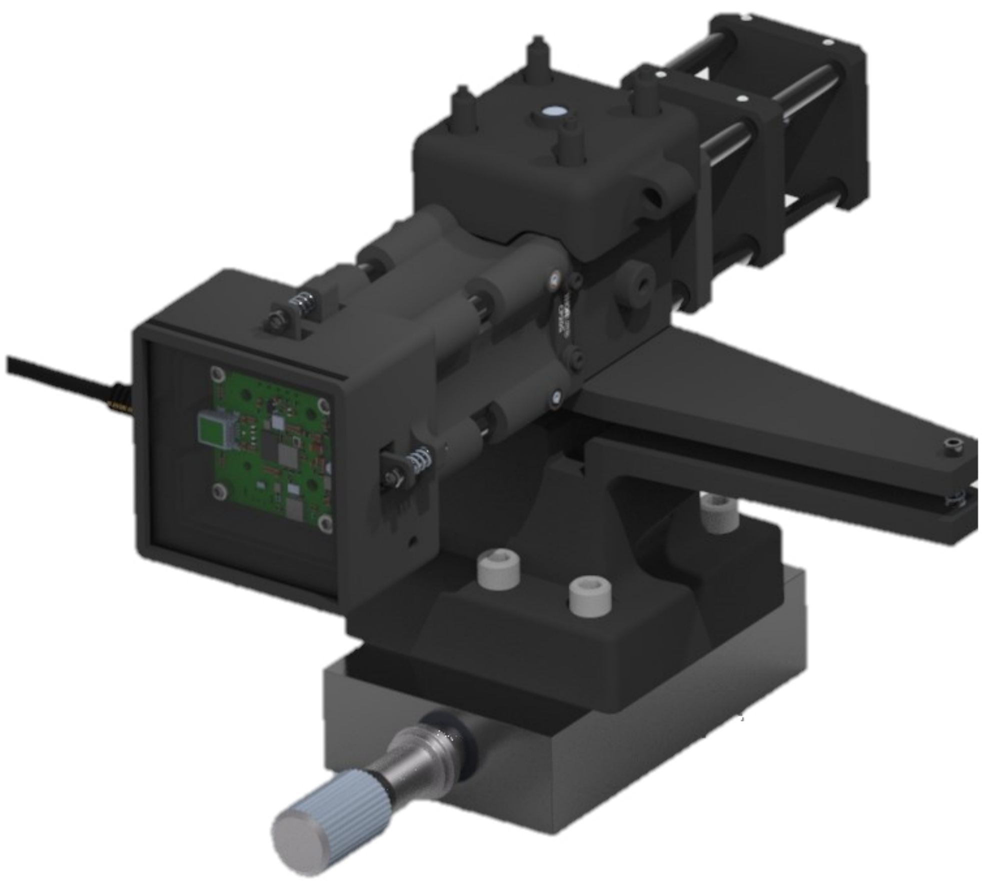


Figure 2.1: Camera assembly

A combination of optical magnification and image processing are required in order to achieve the desired 10 nm resolution. Optical image magnification is chosen to achieve a spacial resolution of 20 nm per image pixel. The camera sensor used for this system, has a pixel size of 1.67 µm, and as a result the required optical magnification necessary to achieve a spatial resolution of 20 nm per image pixel is approximately 85X. After the image has been magnified the sub-pixel routine is then employed in order to achieve the final system resolution. At this spatial resolution, a sub-pixel routine needs to only achieve half-pixel resolution in order to achieve the desired 10 nm resolution.

## 2.1 Magnification System Design

The design goals for the magnification system were that it must produce a 85X magnification, while keeping the optical path length to a minimum. To obtain this magnification, a system of two converging lenses and one diverging lens was designed. The magnification system was designed using thin lens and thick lens paraxial models, as well as Zemax OpticStudio ray-tracing software. The system was designed so that the first lens would produce a 10X magnification and the last two lens would produce a combined 8.5X magnification. Together these elements yield the required 85X magnification.

Table 2.1: Final design lens spacing.

| **label** | **distance** |
|---|---|
| $W_d$ | 9.9 mm |
| $D_1$ | 37 mm |
| $D_2$ | 28 mm |
| $Si_2$ | 151 mm |

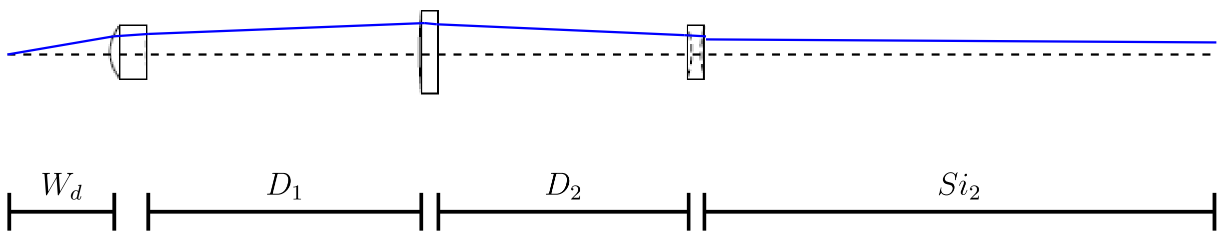



Figure 2.2: Ray diagram

Table (2.2) shows the variable definitions used in the thin lens paraxial model.

Table 2.2: Thin lens model variable definitions.

| **label** | **distance** |
|---|---|
| $W_d$ | working distance |
| $Si_n$ | image distance for the $n^{th}$ lens |
| $S0_n$ | object distance for the $n^{th}$ lens |
| $F_n$ | focal length for the $n^{th}$ lens |
| $M_n$ | magnification produce by the $n^{th}$ lens |

The requirements for the first lens are that it must produce a 10X magnification at a reasonable working distance $W_d$. The magnification produced by the first lens $M_1$ is determined by equation (2.1 )

$$M_1 = -\frac{Si_1}{W_d} \tag{2.1}$$

The relationship between the working distance $W_d$ and the image distance $Si_0$ is determined by equation (2.2)

$$\frac{1}{F_1} = \frac{1}{Si_1} + \frac{1}{W_d} \tag{2.2}$$

Equation (2.1) is substituted into equation (2.2), and solved for $W_d$ which yields equation(2.3). The desired magnification $M_1$ is input as well as the focal lengths of standard "off the shelf" lenses until a configuration resulting in a working distance suitable to the design is achieved.

$$W_d = \frac{F_1(M_1 - 1)}{M_1} \tag{2.3}$$

The resulting image distance from the first lens element can then be calculated using equation (2.4).

$$Si_1 = \frac{(W_d)F_1}{W_d - F_1} \tag{2.4}$$

The image from the first lens serves as the object for the second lens, and therefore the object distance for the second lens is dependent on the distance between the two lenses $D_1$ as seen in equation (2.5).

$$S0_2 = D_1 - Si_1 \tag{2.5}$$

The image distance for the second lens can be calculated as a function of distance $D_1$ using equation (2.6).

$$Si_2 = \frac{(D_1 - Si_2)F_2}{(D_1 - Si_2) - F_2} \tag{2.6}$$

Once the object distance and the image distance for the second lens element is calculated, it is possible to determine the magnification produced by this lens element using equation (2.7).

$$M_2 = \frac{-Si_2}{So_2} \tag{2.7}$$

The combined magnification of the final two lenses can be computed using the equation (2.8)

$$M_2 M_3 = M_{2:3} \tag{2.8}$$

Equation (2.8) can be solved for the magnifcation $M_3$ required for the last two optics to produce the required combined magnification $M_{2:3}$. This magnification can then be used to generate equation (2.9) which determines the separation distance between the second and third lens $M_3$

$$D_2 = \frac{F_3(\frac{M_{2:3}}{M_2}) - 1}{(\frac{M_{2:3}}{M_2})} + \frac{(D_1 - Si_2)F_2}{(D_1 - Si_2) - F_2} \tag{2.9}$$

The object distance for the third lens $So_3$ can be computed using equation (2.10). This distance is the distance between the last lens and the camera sensor.

$$\frac{So_3 F_3}{So_3 - F_3} \tag{2.10}$$

After determining the necessary spacing, $D_2$, the required object distance, $So_3$, and image distance, $Si_3$, are computed in the same manner as for the second lens. At this point, the magnification of the last lens, $M_3$, is calculated. Using equations (2.3), (2.9), and (2.10) the total length of the magnification system can be computed solely as a function of the required magnification, the focal lengths of the lenses and the distance between the first and second lenses. A variety of combinations of focal lengths for "off the shelf" lenses, as well as values for the distance $D_1$ were evaluated using these equations to choose a configuration which produced a suitably compact magnification system. Table 2.3 shows the resulting lens choices.

Table 2.3: Lens properties.

| **Lens** | **focal length** | **material** | **part number** |
|---|---|---|---|
| 1 | 9.07 mm | molded acrylic | APL0609-A(Thorlabs) |
| 2 | 90 mm | N-BK7 | 67157(Edmunds Optics) |
| 3 | −9 mm | N-Bk7 | 48935(Edmunds Optics) |

Once the initial design was established using thin lens paraxial equations, thick lens paraxial equations were used to improve the quality of the model. Following this the design was input into Zemax OpticStudio ray-tracing software to simulate the performance of the system. Small adjustments to the design were made based on the Zemax simulation to reduce aberrations in the system. The design incorporates an adjustable aperture behind the lenses to reduce spherical aberration. To further correct spherical aberrations in the system, an "off the shelf" apsheric objective lens is utilized. The system does not require color imaging, so a narrow bandwidth green LED light source is utilized for the illumination system. The use of a small frequency range of light means that there is little need for achromatic doublets or other elements to correct the chromatic aberrations [9].

Most of the remaining abberations in the system are spherical in nature. The spherical abberation is of much greater magnitude than that of the other abberations in the system as is shown in Figure (2.3). Some of the spherical aberration is however reduced further than what is depicted in the diagram, because the aperture in the physical system is of a smaller diameter than in the Zemax simulation. The Zemax simulation utilizes the largest possible size of the aperture but the final size of the aperture was not known at the time of simulation, because it was manually optimized after construction.

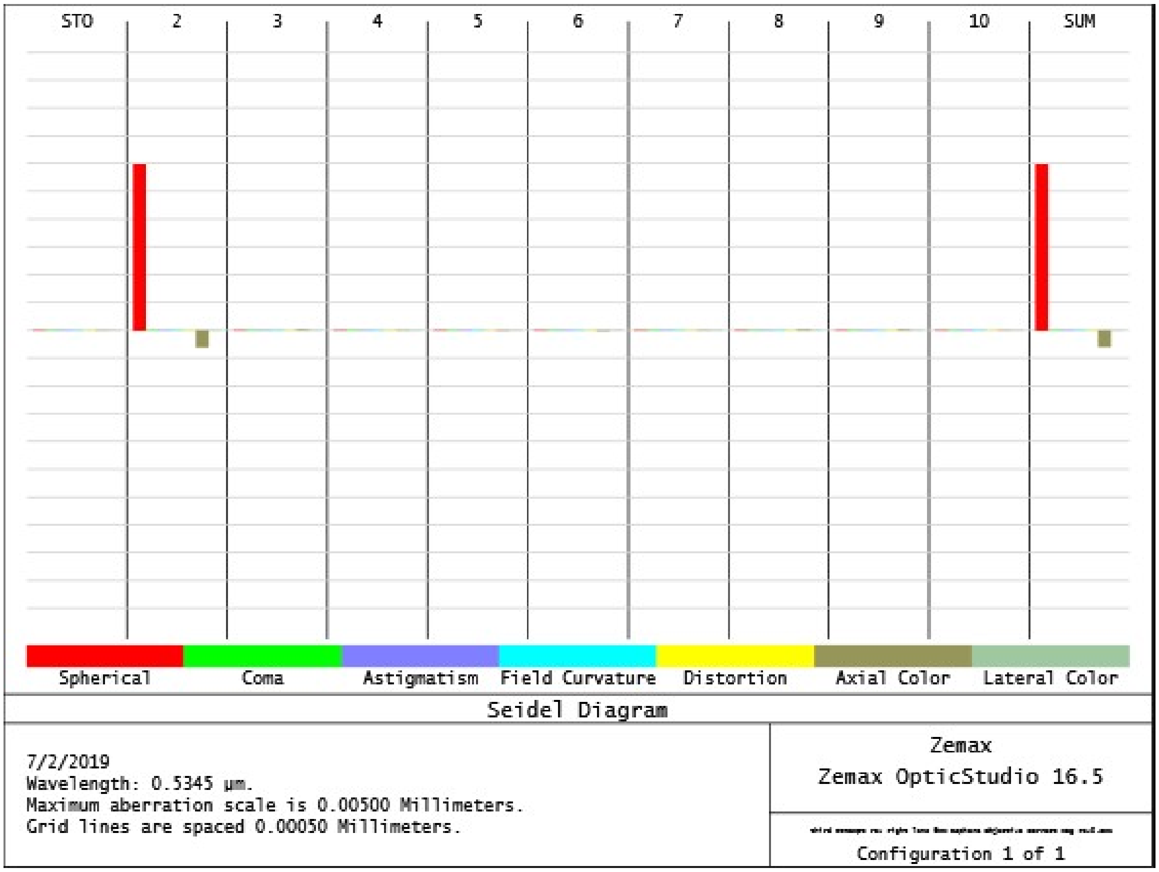



Figure 2.3: Seidel diagram

As previously stated, the design of the optical magnification system is driven by a desire to limit the total length of the system as well as the total cost. Subsequently the system utilizes relatively few optical elements which are all "off the shelf". Meeting these conditions comes at the cost of reduced optical performance, and consequently the magnification optic's performance falls short of being diffraction limited. The optics do, however, provide sufficient resolution and contrast in order to recognize scratches and surface features on the wire necessary to track its movement using DIC. The magnification of the system is such that if the system could achieve the necessary resolution, one pixel would represent 20 nm on the wire. This system actually resolves about 10 lines per mm as shown in the modulation transfer function in figure (2.4), and thus it can resolve approximately $2 \times 10^{-4}$ lines per pixel with its current magnification. While this is significantly below the diffraction limit, it is still sufficient because cross-correlation's sensitivity to image blur is quite low, as is shown in section (7.1).

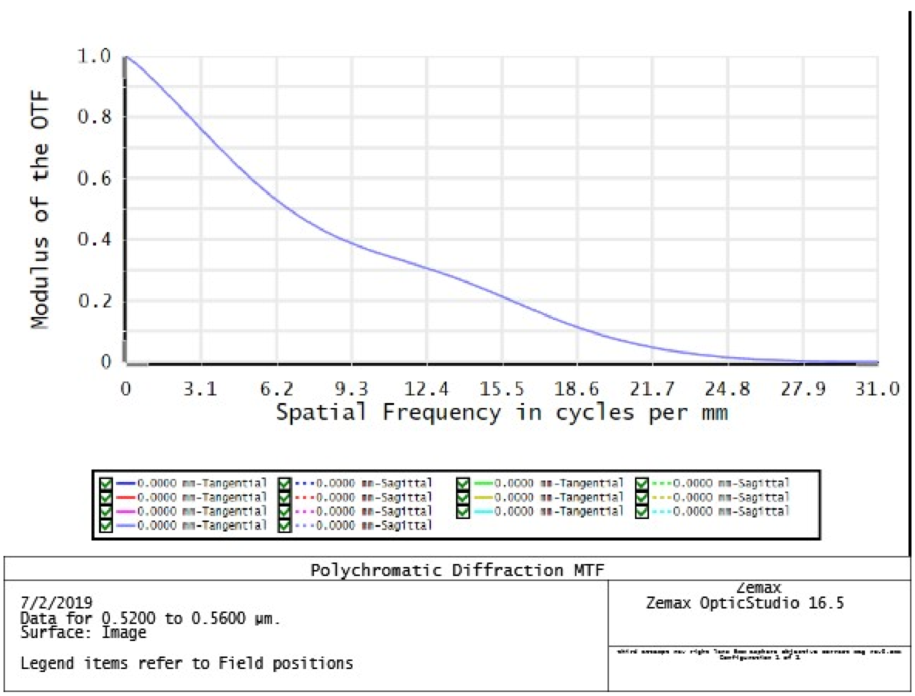
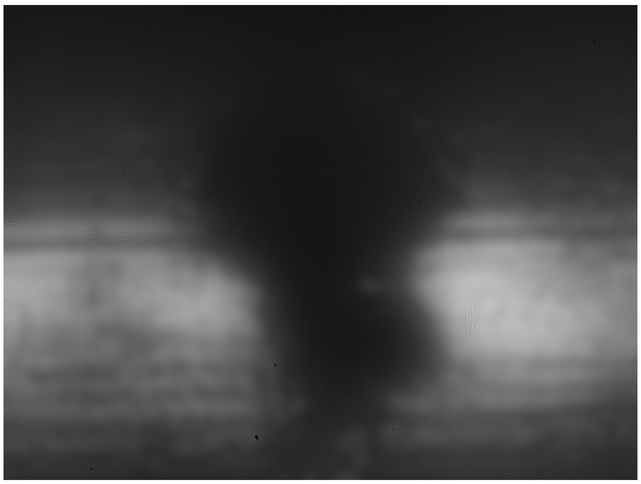



Figure 2.4: Modulation transfer function

Figure (2.5a) shows an image captured by the optics in the instrument. Figure (2.5b) shows an image of a similar wire captured by an Olympus BX51 microscope, utilizing an Olympus UC30 digital microscope camera for image capture. The resolution produced by the instrument's optical system, is on the order of what is expected based on the simulated MTF.



(a) System Microscsope

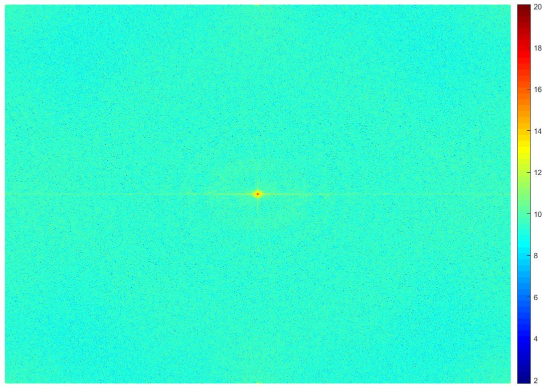
(b) Olympus Microscope

Figure 2.5: Microscope images

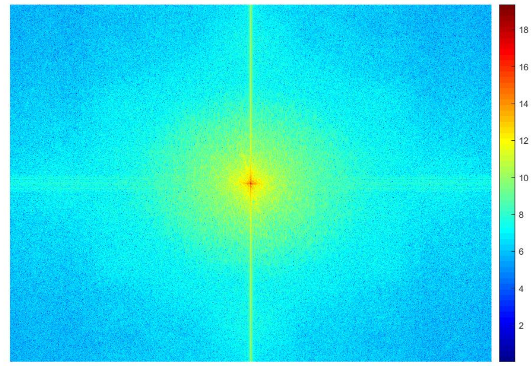
Table 2.4: Microscope specifications.

| | **Instrument microscope** | **Olympus microscope** |
|---|---|---|
| camera model | UI-1492LE-M | UC30 |
| resolution | 3840(H) x 2748(W)pixels | 2080(H) x 1544(W) pixels |
| pixel size | 1.67 x 1.67 µm | 3.45 x 3.45 µm |
| image magnification | 85X | 100X |
| sensor type | monochrome CMOS | color CCD |

Figure (2.6a) shows the FFT of the image from figure (2.5a). Figure (2.6b) shows the FFT of the image from figure (2.5b). From these images we can see that the images captured by the current system does not capture a significant portion of the high frequency content in the image captured by the Olympus microscope.



(a) System Microscsope FFT



(b) Olympus Microscope FFT

Figure 2.6: Microscope image FFTs

The path of light from the test wire to the camera sensor is shown Figure (2.7)

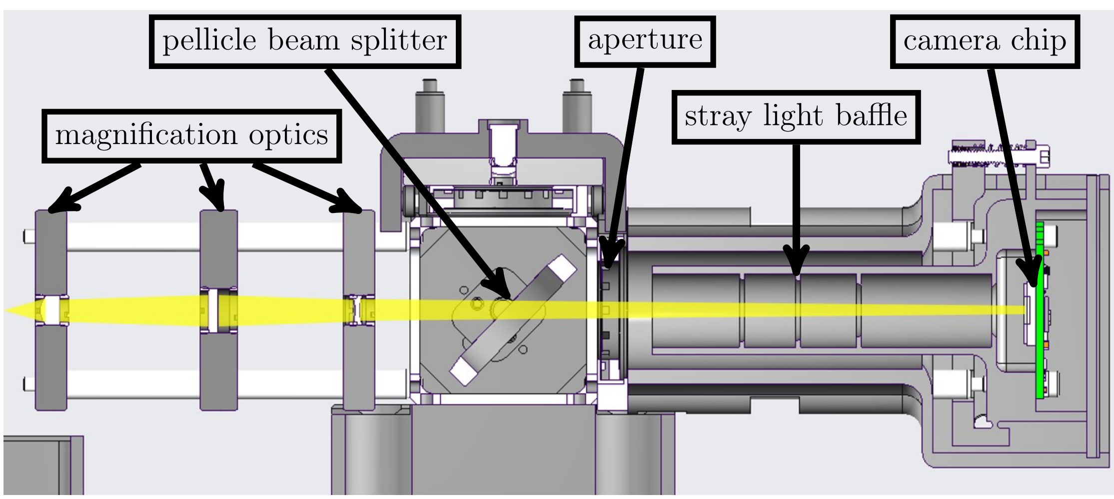



Figure 2.7: Image path

The camera was purchased in the form of an image sensor on a printed circuit board. This form factor allowed the housing system to be designed to meet practical requirements as well as allowing for the purchase of a monochrome camera. The use of a monochrome camera means there is no Bayer color filter to account for in the recorded image. The camera housing was additively manufactured using ABS plastic and incorporates a two-axis flexure system. This flexure system allows fine adjustment of the alignment between the image sensor and the optics. After the light passes through the magnificaiton optics it passes through a pellicle beam splitter, whose purpose will be further expalained in chapter (3). The camera housing also includes an optical baffle which eliminates stray light outside of a 30 degree inclusion angle. The optical baffle is designed to eliminate first and second order reflections within the system. The entire camera and optics assembly is mounted to a flexure system which allows for the fine adjustment of the angle of the camera to ensure that its horizontal axis is well aligned with the axis of travel of the wire. The fine adjustment flexure is shown in figure (2.8). This whole assembly is mounted on top of a micrometer stage which is used to adjust the focus of the system.

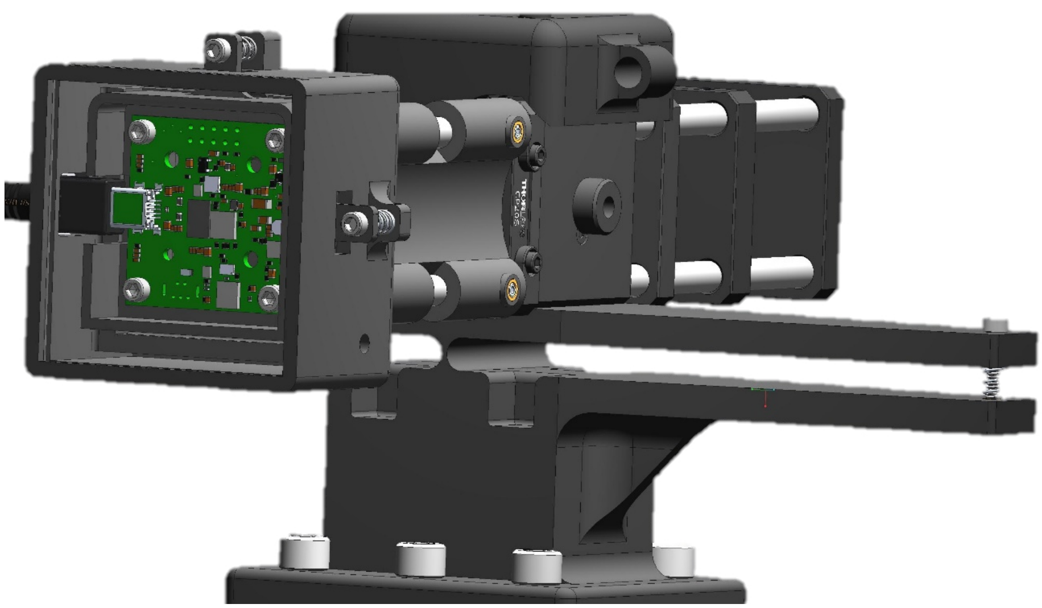


Figure 2.8: Camera fine adjustment flexure

# CHAPTER 3: ILLUMINATION SYSTEM

Since the system is imaging an opaque object, it is not possible to pass illumination light through the object as is normally done with a backlit microscope. In order to illuminate the wire sample, a coaxial illumination system was created which consists of an LED light source, an aperture, a converging lens and a pellicle beam splitter. The aperture limits the angle of the cone of light which can enter the system and thus minimizes scattered illumination light from reflecting into the sensor. The converging lens converts the expanding cone of light from the LED into a nearly collimated light source. The pellicle beam splitter reflects a portion of the light forward into the magnifying optics, which in this case, act to focus the light to a spot 50 percent larger than the field of view of the optics [10]. The remainder of the light which passes through the system hits a surface which has been painted optically black to prevent the light from passing back through the system and into the sensor. The path of light from the illumination LED to the test wire is shown in Figure (3.1).

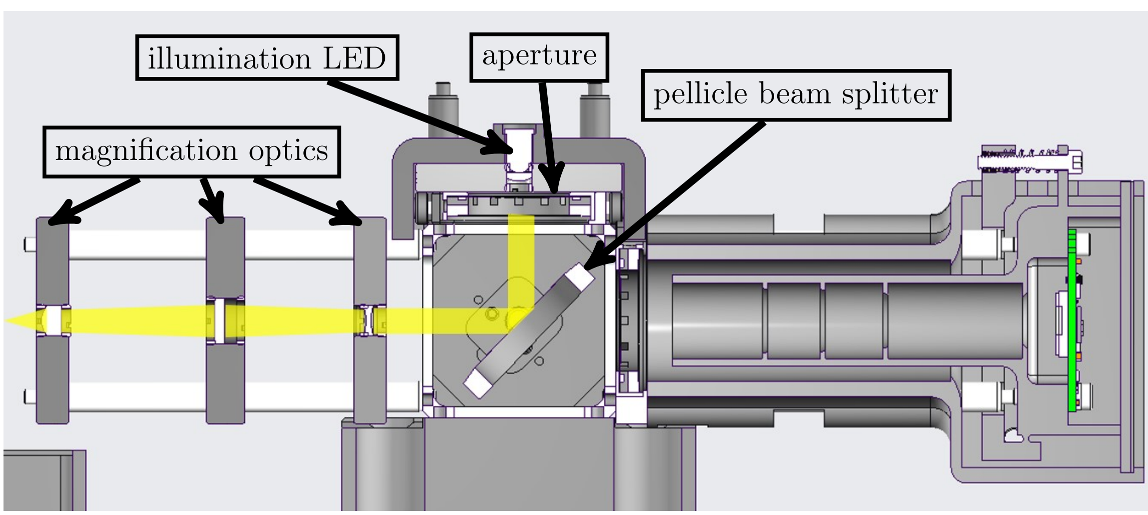



Figure 3.1: Illumination path

## 3.1 Stray Light Control

The housing that contains the camera sensor is 3D printed from ABS plastic and contains an integral stray light baffle with a calculated exclusion angle of 30 degrees. The initial images from the system appeared over-saturated and also showed a large dependence on the ambient illumination in the laboratory. It was determined that light from outside the system was being transmitted through the plastic housing and striking the sensor. When the housing was designed, it was believed that it would shield the sensor from the majority of outside light. In practice however, it appears as though small amounts of light were able to pass through the small voids which resulted from incomplete bonding of print layers. To reduce this effect the housing was painted optically black. Painting the camera housing both served to help fill any small voids where light might be able to enter the housing, as well as reduce internal reflections. It was also discovered that light was transmitting through the rear of the camera sensor chip, and subsequently a rear plate was included in the system to better enclose the camera. To further prevent unwanted stray light from entering the system, the entire system was surrounded by an opaque environmental isolation enclosure. The inclusion of this enclosure also significantly reduced disturbances to the system caused by irregular air flow within the testing environment. Another source of stray light within the system is the illumination LED. A portion of the light from the LED scatters when it hits the pellicle beam splitter and eventually reflects into the image sensor. To help mitigate this issue, a simple ray tracing program was written which would track the marginal ray of the illumination light. Using this ray tracing program, the previously mentioned converging lens was selected which helps to narrow the path of the light from the LED. Both the focal length of this lens and its distance from the LED were calculated to produce an illumination spot size approximately 2 times the diameter of the wire once it passes through all of the magnification optics. [11]. The lens which was selected has a focal length of 10 mm, and is placed at a distance of 0.12 in from the tip of the LED.

# CHAPTER 4: WIRE HOLDER ASSEMBLY

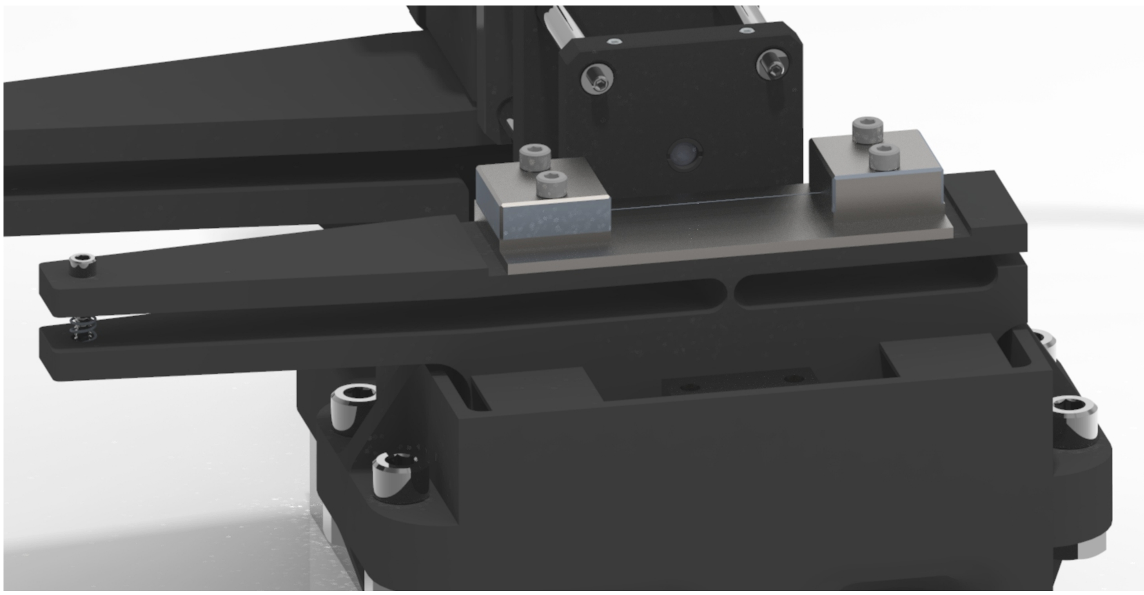


Figure 4.1: Wire holder on rotational flexure with screw adjustment

The sample wire and the linear encoder glass scale are mounted to a platform which incorporates two separate flexure mechanisms. The first flexure mechanism allows for fine rotational adjustment to align the wire with the linear encoder. This flexure angle is adjusted using a screw and spring system. The flexure and its adjustment system is shown in figure (4.1). The spring preloads the flexure and the screw at the end of the adjustment lever is then used to control the rotational position. The spring must be stiff enough to rotate the flexure far enough in the clockwise direction to contact the stop. For this to be the case, the spring and the flexure must be in rotational equilibrium at an angle greater than or equal to the angle $\theta_{max}$ at which the flexure will contact the physical stop.

A maximum rotational range ($\theta_{max}$) of 3° was selected in order to allow sufficient range to properly align the flexure with the camera. The sum of moments is presented in equation (4.1), which is used to calculate the equilibrium point between the coil spring and the flexure.

$\theta_{max}$ = maximum rotation of notch flexure

$k_1$ = rotational stiffness of notch flexure

$k_2$ = linear stiffness of coil spring

$r$ = length of lever arm from flexure hinge to adjustment screw

$\Delta y$ = change in length of coil spring

$$\sum_{moments} = 0 = k_1\theta_{max} - k_2\Delta_y r \tag{4.1}$$

Equation (4.2) is then used to calculate the change in length of the coil spring, as a function of the maximum rotational position of the flexure hinge. The diagram in figure (4.2) shows the relations which were used to calculate the change in coil spring length as a function of flexure rotation.

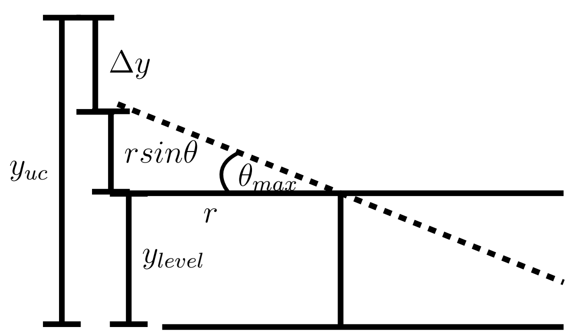



Figure 4.2: Relations for spring deflection and flexure rotation

$y_{uc}$ = uncompressed length of coil spring

$y_{level}$ = length of the coil spring when the flexure is level

$$\Delta_y = y_{uc} - y_{level} - rsin(\theta_{max}) \tag{4.2}$$

$t$ = minimum thickness of notch flexure

$b$ = depth of notch flexure

$a_x$ = radius of notch flexure

$E$ = Young's modulus of flexure material

$k_1$ = rotational stiffness of notch hinge flexure

Equation (4.3) is used to calculate the rotational stiffness of the flexure hinge [12].

$$k_1 = \frac{1}{\frac{3}{2Eba_x^2[\frac{1}{2\beta+\beta^2}]}\left[\left(\frac{2+4\beta+2\beta^2}{(1+\beta)(2\beta+\beta^2)}\right) + \left(\frac{6(1+\beta)}{(2\beta+\beta^2)^{\frac{3}{2}}}\right) tan^{-1}\sqrt{\frac{2\beta}{\beta}}\right]} \tag{4.3}$$

Where

$$\beta = \frac{t}{2a_x} \tag{4.4}$$

Substituting equation (4.2) into equation (4.1), and applying the small angle approximation, yields equation (4.5) which gives the required stiffness of $K_2$.

$$K_2 = \frac{ry_{uc} + ry_{level} - r^2\theta_{max}}{-k_1\theta_{max}} \tag{4.5}$$

These equations were used to choose appropriate geometry for the flexure hinge, as well as the coil spring stiffness. With $b$=0.025 in, $t$=0.05 in, and $a_x$=1 in, the rotational stiffness $k_2$= 0.23 $\frac{\text{lb} \cdot \text{in}}{\text{degree}}$.

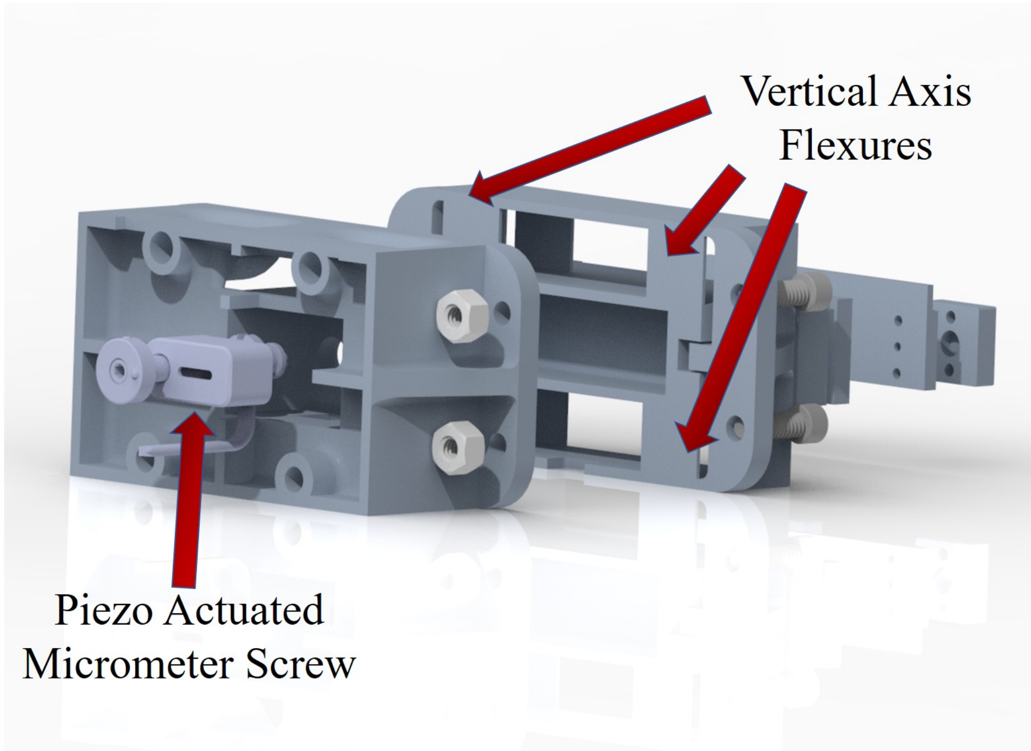



Figure 4.3: Vertical axis flexure and piezo actuator

The second flexure mechanism, which allows the wire sample to be positioned vertically, is shown in figure (4.3). The vertical flexure system incorporates a digitally controlled actuator which uses a piezo to generate stick slip rotational motion. This rotational motion is imparted into a micrometer screw in order to generate precision linear movement. This actuator is called a Picomotor. A photograph of a Picomotor is shown in figure (4.4). The Picomotor allows for precise control of the wire's vertical position with respect to the camera. The vertical flexure system consists of 8 total flexures, two sets near the the top of the wire holder and two sets near the bottom. The flexures were positioned so as to limit the rotation of the system due to any moments potentially generated if the system is driven slightly off axis.

Equation (4.6) is used to compute the stiffness of each flexure element [12].

$$k_3 = \frac{12EI}{L^3} \tag{4.6}$$

A flexure width of 1.5 in, a length of 0.79 in, and a thickness of 0.04 in results in a stiffness $K_3$ of 1.13 $\frac{\text{lb}}{\text{in}}$ per flexure element and a total stiffness of 9 $\frac{\text{lb}}{\text{in}}$. A displacement of 4.4 mm requires 7 N. This displacement range is sufficient to track the vertical position of the wire, and the force required to produce this displacement is well below the 22 N the Picomotor is capable of producing.

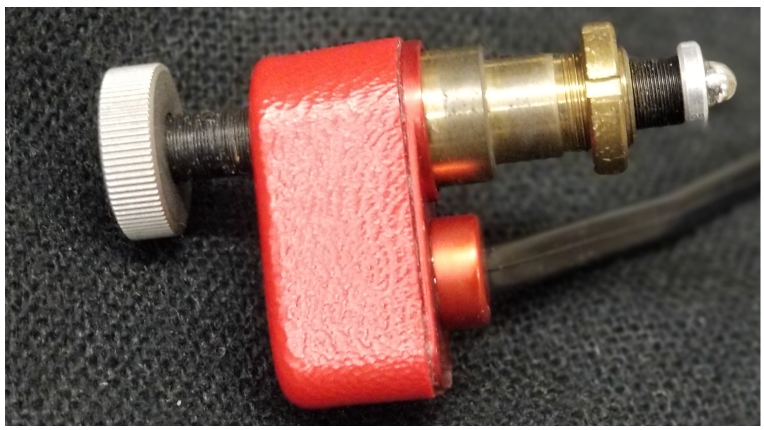


Figure 4.4: Picomotor

Table 4.1: Picomotor specifications

| **specification** | **value** |
|---|---|
| travel range | 25.4 mm |
| minimum motion | $<$ 30 nm |
| max speed | 1.2 mm min$^{-1}$ |
| max force | 22 N |

By using the same material for both the camera mount and the wire holder, their thermal expansion is well matched. This helps minimize differential thermal expansion between the two assemblies. In addition, both the camera mount and the wire mount have been designed to be close to symmetrical about their vertical axis. Neither the camera mount, nor the wire mount are symmetrical about its horizontal axis, which means that there is appreciable differential drift in the vertical direction between the assemblies. To compensate for this effect, the vertical flexure stage's position is held constant through closed loop feedback from the camera image. In this system, the wire and encoder are translated while the camera is held in a fixed position. The wire mount assembly, as well as the encoder glass scale, are mounted on top of a translation stage which is actuated by a picomotor. Mounting these components together allows for the measurement of the wire translation using the encoder system.

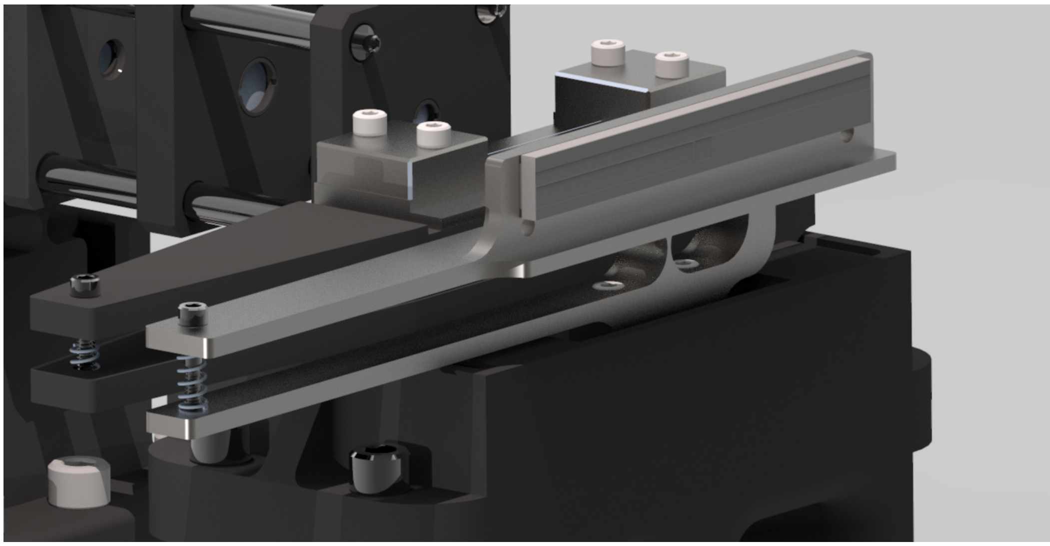


Figure 5.1: Encoder flexure

This instrument utilizes a Heidenhain LIF 401R scale, a LIF 48 encoder read head, and a proprietary electronic interface box. The encoder glass scale is mounted to the same translation stage as the wire holder; however, it does not physically contact the wire holder assembly. The steel mounts for the linear encoder's glass scale and the linear encoder's read head are manufactured from the same batch of steel to help minimize differential thermal expansion which could affect the alignment of the encoder scale and its read head. The glass scale mount incorporates a flexure assembly which allows for the fine rotational alignment of the encoder scale to the encoder read head. This flexure is shown figure (5.1).The surface of the encoder read head must be placed $1\,\mathrm{mm} \pm 0.01\,\mathrm{mm}$ relative to the front surface of the glass scale, with parallelism of $\pm 0.06\,\mathrm{mrad}$. In order to achieve this, the structure to which the glass scale and the encoder read head are mounted to are designed with a registration surface. This surface allows the precision alignment of the components with a dial indicator without actually contacting the encoder or glass scale. The glass scale mount includes precisely machined holes, into which 0.125 in dowel pins can be positioned so as to support the encoder in the proper alignment as it is rotated into the correct position.

# CHAPTER 6: SOFTWARE

The software for this instrument initializes the camera and captures an image from it, performs cross-correlation to determine the location of a set of features on the wire, controls the vertical and horizontal axes of the wire's translation, reads and outputs position measurements from the linear encoder, and records temperature measurements near the sample wire. In many cases, the software required for the image processing and hardware interfacing is very computationally intensive. Stemming from this need for computational efficiency, C++ was chosen as the programming language for use in this instrument because it is quite low level and allows for relatively straight forward optimization. Since most of these processes must occur concurrently, multi-threading has been incorporated into the software to create multiple execution streams using the thread architecture depicted in figure (6.1). A LabJack DAQ system is used to interface the control PC with the Picomotors and temperature sensor. A proprietary electronics interface box manufactured by Heidenhain, is being used to interface the control PC with the linear encoder. The digital camera has an integral USB interface to communicate with the control PC.

## 6.1 Multi-Threading

It is necessary that many of the components of this system such as the camera, encoder, and LabJack DAQ system run concurrently and asynchronously when the system is in operation. In order for these systems to run continuously, they must run inside a continuous loop. This would be impossible using a single execution stream architecture, so a multi-threaded program structure is utilized. In a multi-threaded system, the main thread spawns additional sub-threads. Each additional sub-thread, executes independently from and simultaneously with the other threads. Because the individual threads must often share information between themselves, great care must be taken to prevent what is known as a race condition. A race condition is where one thread requests information from another thread before it has had adequate time to complete its own processing task. The use of multi-threading also allows for the control computer to make more efficient use of its multiple processors, by completing computational processes partially in parallel, rather than entirely in series. A diagram of the data flow within the multi-threaded structure is shown figure (6.1).

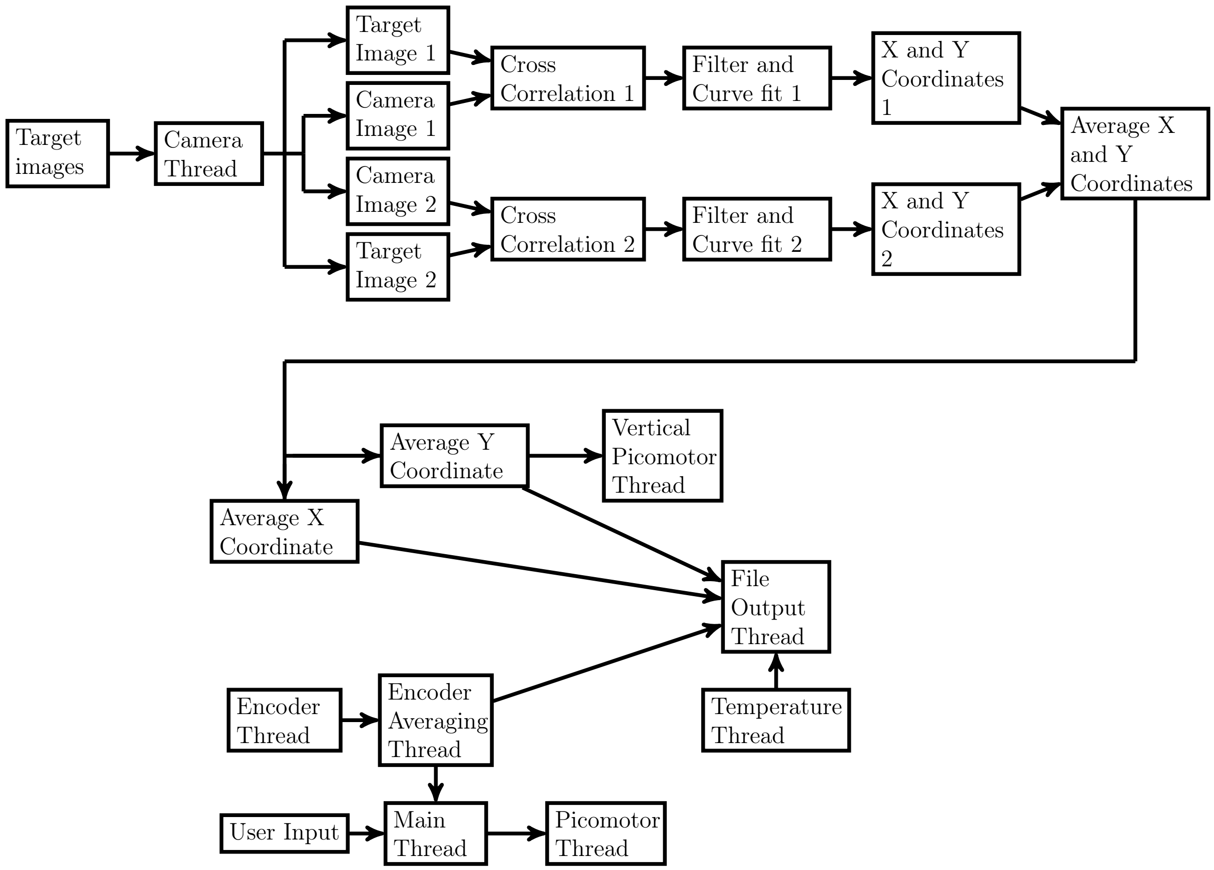



Figure 6.1: Simultaneous thread operations

### 6.1.1 Main thread

Upon software start up, the main thread first initializes the connection to the LabJack DAQ system. Once this connection is established, the main thread begins the creation of sub-threads. The next operations for the main thread are to connect, initialize, and begin receiving position data from the encoder. The encoder sends absolute position data so an offset is subtracted from the position data in order to ensure the initial position of the system is zero. Next, it prompts the user to ask if they would like to continue. If the user chooses to do so, they are next prompted to input a desired location on the wire in nanometers. After inputing this information, the main thread calculates the difference between the system's current position and the desired position $\Delta x$. If $\Delta x$ is greater than the specified maximum acceptable error, the program commands the picomotors to take one step in the appropriate direction. From this point, the main thread recomputes the $\Delta x$. If $\Delta x$ is greater than the specified maximum acceptable error value, the program continues through the loop. If the $\Delta x$ value is less than the maximum acceptable error value, the program breaks out of the loop and again prompts the user if they would like to continue. If the user chooses not to continue, the main thread will exit the camera feed and release its memory location. At this time all of the sub-threads rejoin the main thread.

### 6.1.2 Camera thread

In this thread, Target reference of the desired feature sets on the wire, are imported. Once the target images are imported, communication to the camera is established, camera parameters are set, and the camera begins capturing a continuous stream of images. It is inside this thread that the captured images are compared against the target references in order to produce a correlation coefficient matrix. At this time a function is employed to find the location of the maximum correlation coefficient. The location of the maximum correlation is displayed to the screen and the location of the best fit, designated by a red square, is superimposed over the live camera stream. This process occurs for two separate image locations.

### 6.1.3 Measurement averaging thread

In order to reduce the effect of any non-uniform phenomena within the image, two separate cross-correlation processes occur with two target regions. These two measurements are then averaged as illustrated in equation (6.1) and figure (6.2). This averaging effect also helps to reduce the effect of higher local noise levels in dark portions of the image.

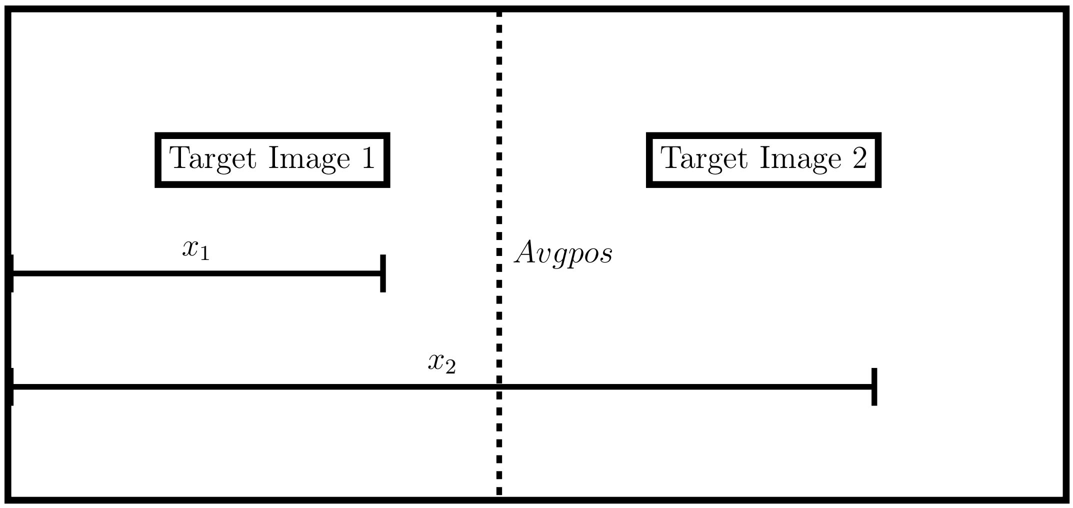



Figure 6.2: Averaging of multiple image locations

$$x_{avg} = \frac{x_1 + x_2}{2} \tag{6.1}$$

### 6.1.4 Encoder thread

The encoder thread initially establishes a TCP-IP connection to the encoder electronic interface box (EIB). Once the connection is established, the EIB system performs an automatic system check. If the system is functioning properly, the program enters a loop which continuously polls position information from the encoder/EIB system.

### 6.1.5 Encoder averaging thread

Values from the encoder thread are sent to the encoder averaging thread where they are averaged over a specified number of iterations. This average measurement value is then sent to the main thread to compute the position and $\Delta x$ value, as well as to the file output thread.

### 6.1.6 Temperature thread

The temperature sensor input thread continually polls the analog port on the LabJack which connects to a LMT-87 analog temperature sensor located near the wire. The voltage signal is processed through a function, which converts it into temperature in °C.

### 6.1.7 Picomotor thread

The picomotor thread takes commands from the main thread which indicate the direction and the speed at which the picomotors should step. These commands then switch the appropriate digital pins for the LabJack DAQ system. The digital pins then connect to the driver for the picomotors. Digital pin 1 determines the direction of motion of the horizontal axis picomotor. When pin 1 is high, the picomotor's drive direction is set to rotate in the clockwise direction. Clockwise rotation results in the stage moving to the left (as viewed from the camera). If pin 1 is low the picomotor drive direction is set to rotate in the counter-clockwise direction. Counter clockwise rotation results in the stage's movement to the right. To make the horizontal axis picomotor step, digital pin 2 is commanded high and then low. The picomotor will take a step when the pin goes low.

### 6.1.8 Vertical control thread

The optics and wire holder systems were designed with symmetry about the vertical axis in mind. This allows the expansions within the system to partially cancel. This cancellation helps keep the camera, optics, wire, and encoder well aligned. The aforementioned systems do not, however, have a great deal of symmetry about the horizontal axis, which allows for a great deal of expansion of the system vertically. This poses issues with keeping the wire centered vertically in the camera's field of view. Without vertical correction, the wire can drift completely out of view of the camera. In order to combat this issue, the vertical axis flexure is used to keep the wire centered. When the vertical control thread is initialized, it accesses the vertical position of the wire as determined by image correlation. This initial position becomes the set point for a simple proportional position regulation control system. The control system operates only if the vertical position exceeds a range of $\pm$ 3 pixels. The control system incorporates a fail-safe system in the event that there is a disturbance to the system which causes the camera to lose view of the wire target features. If the wire features go out of view of the camera, the software defaults to reporting a correlation location of 0. If this were allowed to occur, the system could driven in the wrong direction to an unreachable destination. This would cause the system to generate a large stress in the flexure system, and would also produce excessive wear within the picomotor. To avoid this, the system automatically terminates if the location of the target feature is reported to be zero. Digital pin 3 of the Labjack controls the direction of movement for this axis, and pin 4 commands this axis to step. A block diagram of this control system is shown in figure (6.3).

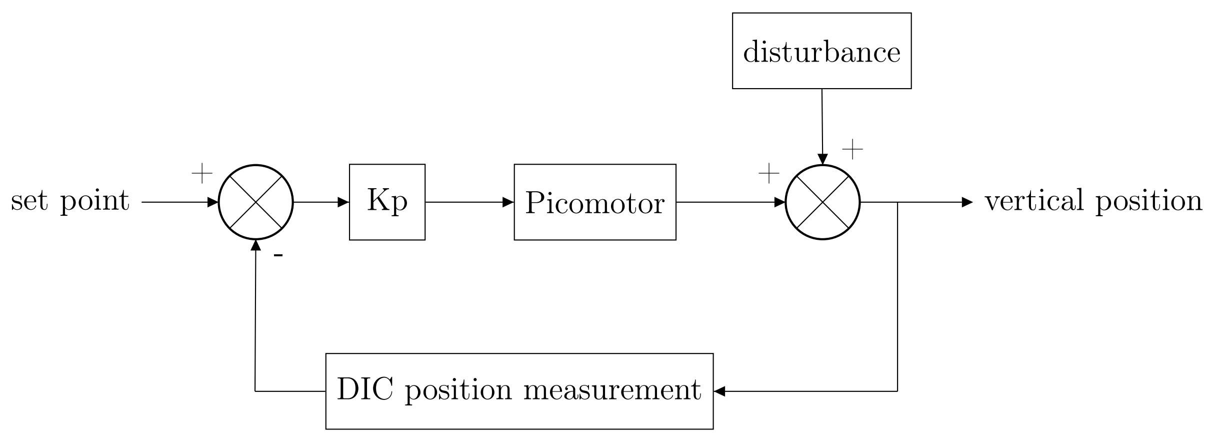



Figure 6.3: Vertical axis control block diagram

# CHAPTER 7: NORMALIZED CROSS-CORRELATION

The image processing method which is employed to track the target feature sets on the wire, is cross-correlation. Cross-correlation is the integral of the conjugate of a shifting function, multiplied by a stationary function.

$$R(t) = f * g = \int_0^t \overline{f(u)} g(t-u)\, du \tag{7.1}$$

One of the large advantages to employing normalized cross correlation for determining the position of a feature set is its relative insensitivity to quality of the optics. To illustrate this point, the case of a simplified 1-D example can be examined. The possible intensity value range of an image is bounded between 0 and 255, and for this reason a sine function with a vertical offset is used to demonstrate this point.

$$f(x) = a\sin(x) + b \tag{7.2}$$

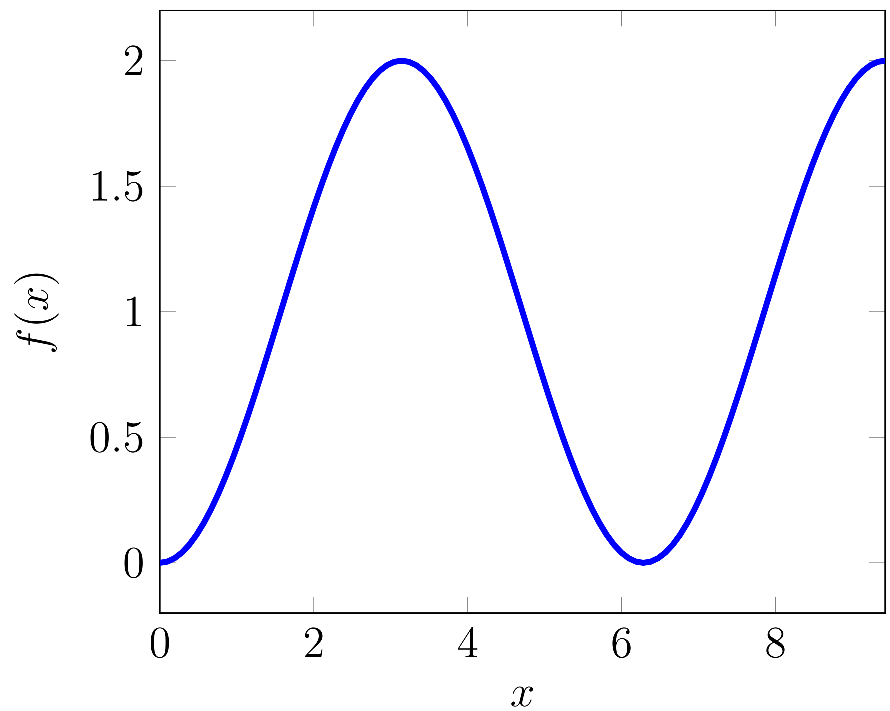



Figure 7.1: Sine function: $f(x) = asin(x) + b$, $a = 1$, $b = 1$

Substituting equation (7.2) into functions $f(u)$ and $g(t-u)$ yields the equation for cross-correlation of the sine wave.

$$R(t) = \int_0^t (a\sin(u) + b)[a\sin(t-u) + b]\, du \tag{7.3}$$

The integration of equation (7.3) leads to equation (7.4)

$$R(t) = \frac{1}{2}a^2(t\sin - t(t\cos)) - 2ab(t\cos - 1) + b^2 t \tag{7.4}$$

The correlation coefficient depends both on the geometry of the sine function as well as the contrast of the image which is a function of $a$. Contrast with regards to a gray-scale image can be thought of as the range of gray intensities between the brightest and the darkest portions of an image. Blur in an image has the effect of reducing the frequency content and contrast of image data. In this example contrast of an image is analogous to the height of the peak to peak distance of the sine wave. This peak to peak height in the sine wave example is determined by the value of the variable $a$. It is shown that if $a = 0, b = 0$, no correlation is possible. In the case where $a = 0$ and $b \neq 0$ the function reduces to $b^2 t$, and all of the sinusoidal components are eliminated from the function. While it is possible to generate a correlation, it is meaningless in the context of matching the functions. Some contrast is therefore necessary in order for the image to have a data signature distinct enough to be correlated. The 1-dimensional analog can be said to have contrast if $f(t) \neq 0$, $g(t) \neq 0$, $f'(t) \neq 0$, $g'(t) \neq 0$ at some point in the function. In the absence of noise in the image, any image meeting these requirements can be perfectly correlated using cross-correlation as long as the image has a contrast level greater than zero. Even in the extreme case where an artificially generated image has only a single gray pixel with all the remaining pixels being white, the single gray pixel can still be located with perfect accuracy. This result is no longer true however, in a real image where noise is present. When the contrast decreases, the signal to noise ratio decreases, resulting in correlation errors. The cross-correlation of two identical finite length sine functions is shown in figure (7.2)

For digital image analysis, the 2-D discreet form of cross-correlation, shown in equation (7.5) must be utilized.

$$R(x, y) = \sum_{x', y'} (T(x', y')) \cdot I(x + x', y + y')) \tag{7.5}$$

The coordinates of the original image are $x$ and $y$, where as the coordinates of the shifting target image are $x'$ and $y'$

The shortcoming of this form of the equation is that it is very sensitive to the magnitude of the functions. To eliminate this issue the normalized form, shown in equations (7.6) is often employed.

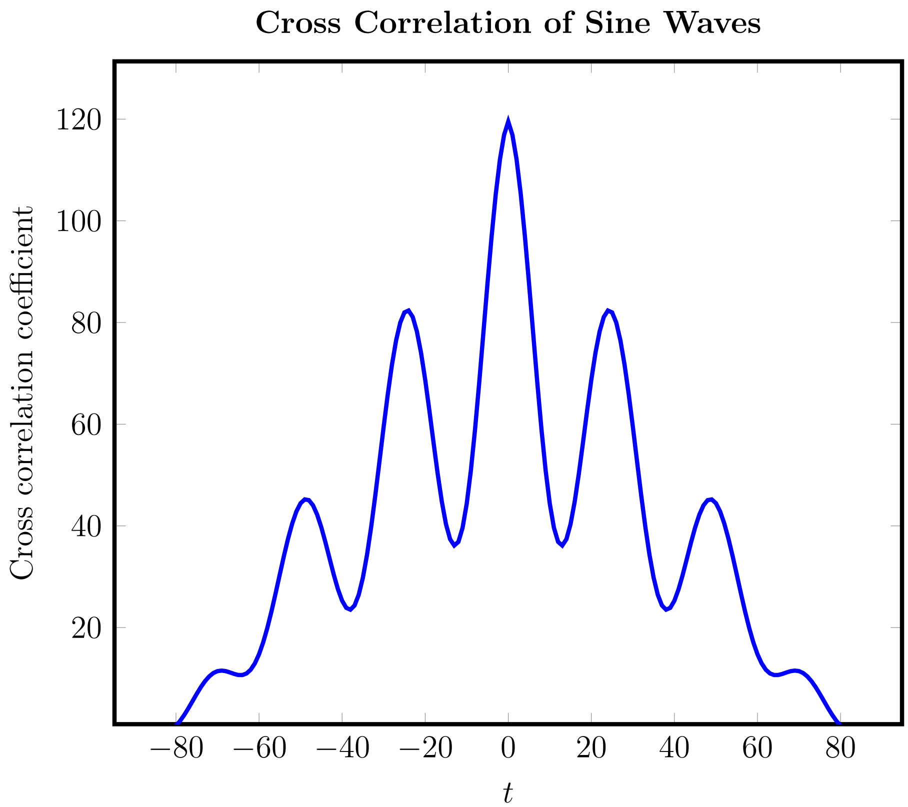



Figure 7.2: The peak of the curve represents where the sine waves completely overlap

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y')) \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \tag{7.6}$$

The normalized form of cross-correlation results in correlation coefficients ranging between $-1$ to 1. A cross-correlation coefficient of 1 represents a perfect match and a value of -1 being an inverse correlation.

## 7.1 Correlation With Noise

Adding noise to a function causes random errors in the location of the maximum correlation coefficient. This effect is particularly significant when there are a relatively small number of samples in the correlation. As the number of samples correlated increases, the error tends toward 0. The higher the signal to noise ratio is, the less the initial error, and the faster it decays when the number of samples is increased. To demonstrate this behavior, Gaussian noise was added to a sine function and then cross-correlated with a noiseless sine function. This model was used because the noise captured by the camera continuously changes, while the noise in the target reference does not. Once correlation of the two functions is complete, the location of the maximum correlation coefficient is determined and recorded. This process is repeated 30 times, and then the standard deviation of these 30 locations is calculated and recorded. After the standard deviation is calculated, the number of samples used in the correlation is increased, and the entire process is repeated for a specified number of cycles.

Figure (7.3) Shows that when the sine function with added noise, has a signal to noise ratio of 1, the standard deviation approaches zero relatively slowly, as the number of samples being correlated increases.

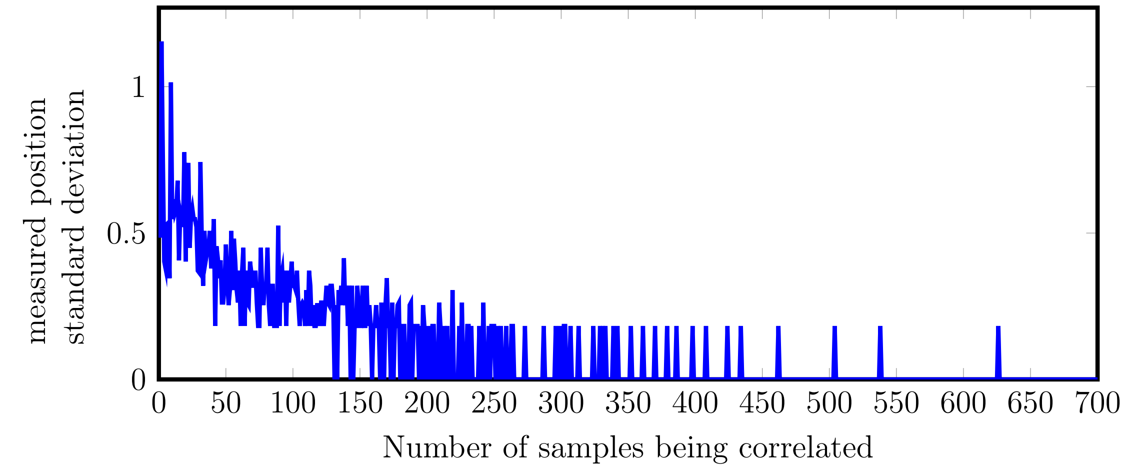



Figure 7.3: Standard deviation vs number of samples being correlated when signal to noise ratio is low

Figure (7.4) Shows that with a larger signal to noise ratio of 3, increasing the number of samples used in the correlation drives the standard deviation to zero more quickly.

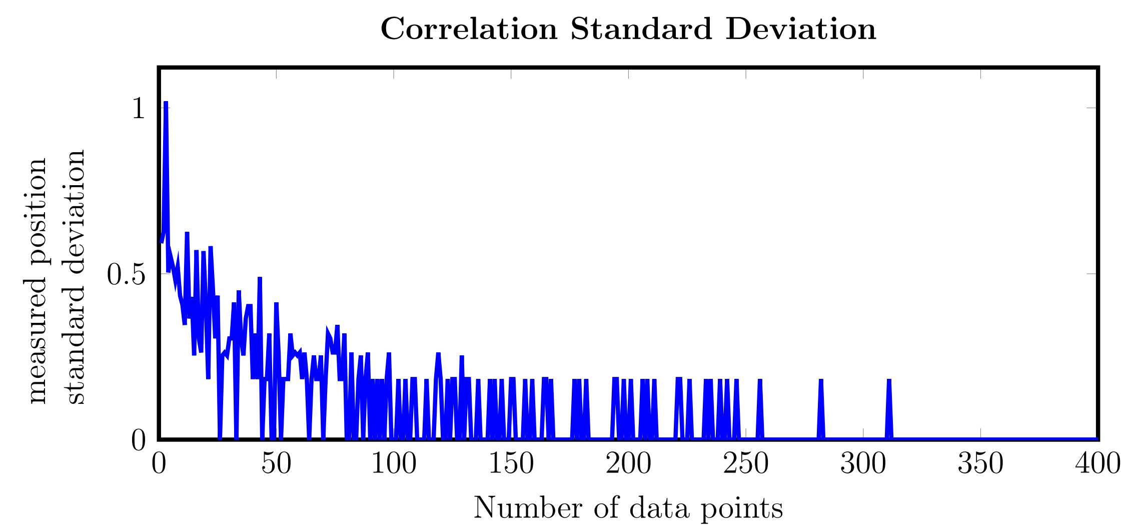



Figure 7.4: Standard deviation vs number of samples being correlated when signal to noise ratio is high

The large amounts of data captured by the camera system help to decrease the standard deviation of the maximum correlation coefficient, which allows the system to maintain single-pixel stability even with a relatively simple optical design. This stability, combined with the estimated $20\,\frac{\text{nm}}{\text{pixel}}$ spatial resolution, enables the system to resolve to 20 nm before sub-pixel image processing.

### 7.1.1 Sharpness vs noise study

In order to further understand the effects of sharpness of the system's optics on the resulting cross-correlation coefficient, a simulation study was conducted where the sharpness of an image captured by the instrument's camera system is reduced by applying a Gaussian filter [13]. Since there is a certain unavoidable amount of noise within the image, it is of interest to determine to which effect the resulting cross correlation coefficient is most sensitive.

First an image captured from the camera system is imported into MATLAB. A region of interest (ROI) section of the image is cropped and stored as a target image. First a Gaussian filter is applied to both images. Next noise is added separately to both the full image and the cropped image. These images are then cross-correlated. This is then done iteratively in a loop, increasing the noise and Gaussian levels with each iteration. A surface fitting routine is then applied to the resulting correlation coefficient surface in order to generate an equation, and the partial derivatives of this equation are calculated in order to determine the sensitivity of the cross-correlation to both parameters.

Figure (7.5) below shows a surface fit representing the simulated cross-correlation coefficients which result from varying the image blur as well as the image noise.

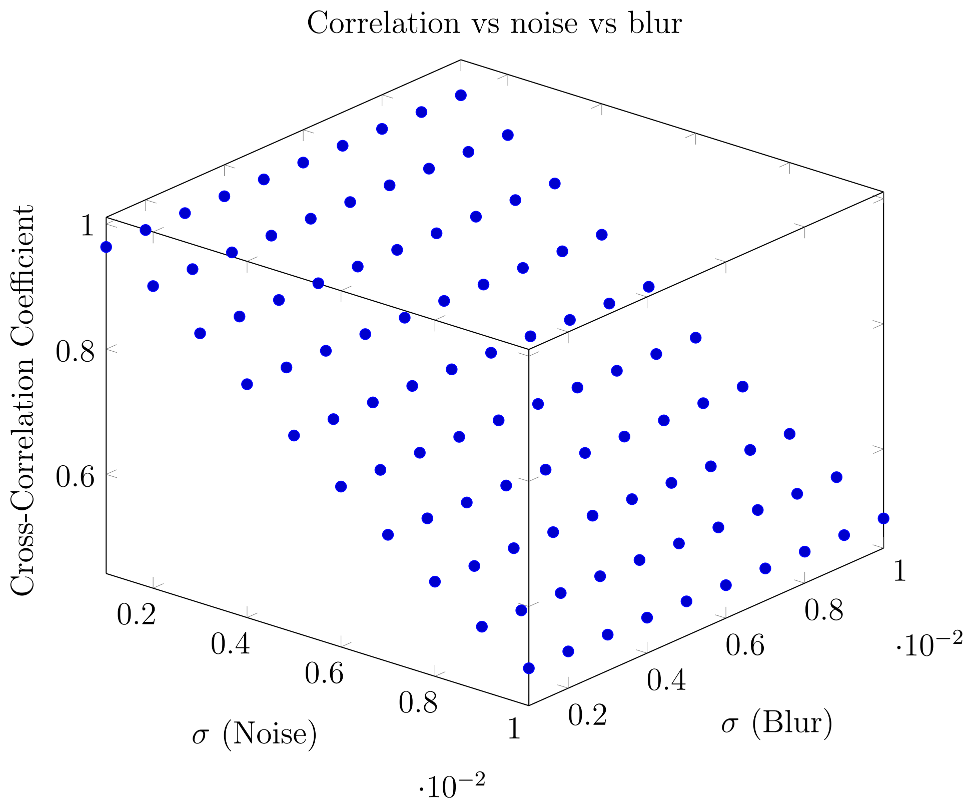



Figure 7.5: Correlation vs noise vs blur

Equation (7.7) represents a polynomial fit to the blur and noise data with a R-squared value of 0.9981 and sum of squared errors value of 0.004543.

$$F(b, n) = 1.027 - (.001437)b - (0.0543)n \tag{7.7}$$

The below partial derivatives with respect to image blur and image noise are linear for both image blur and noise. The partial derivatives also provide evidence that the sensitivity of the cross-correlation coefficient to noise in the image is far greater than that of blurring of the image.

$$\frac{\partial F}{\partial b} = -0.001437 \tag{7.8}$$

$$\frac{\partial F}{\partial n} = -0.0543 \tag{7.9}$$

# CHAPTER 8: SUB PIXEL RESOLUTION

The optical magnification within the instrument produces a spacial resolution of $20\,\frac{\text{nm}}{\text{pixel}}$. In order to achieve the desired $10\,\text{nm}$ resolution, a method to reach sub-pixel resolution has been devised. Normalized cross-correlation is used to compare the live image from the camera to a template image of the target features. This cross-correlation process produces a matrix of correlation coefficients for each corresponding row and column, coordinate pair. For further analysis this matrix can be viewed as a surface. The peaks of the surface represent the areas in the image with the highest correlations to the target image. Initially the surface typically includes both the desired global peak value as well as several local peak values, as is shown in figure (8.1).

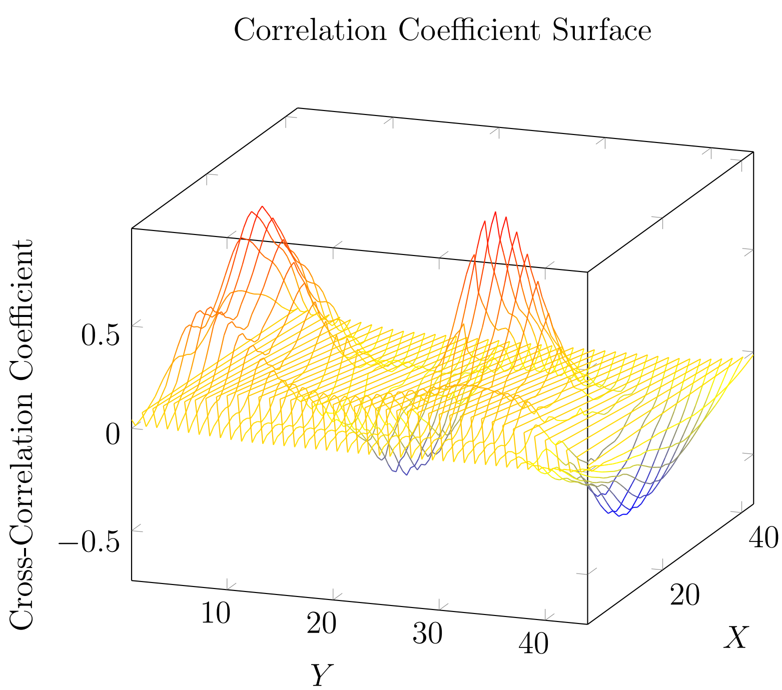



Figure 8.1: Example correlation surface

In order to further process the data, it is necessary to filter it so that only data points representing the global peak are present. To accomplish this filtering, the location of the highest correlation value is calculated, and the data in a surrounding box is cropped from the remainder of the data. This captures the global peak and prevents local peaks from generating erroneous data points. This data is again filtered to reject any points below a specified cross-correlation coefficient threshold. The resulting filtered data is shown in figure (8.2) with a correlation coefficient threshold value of 0.9.

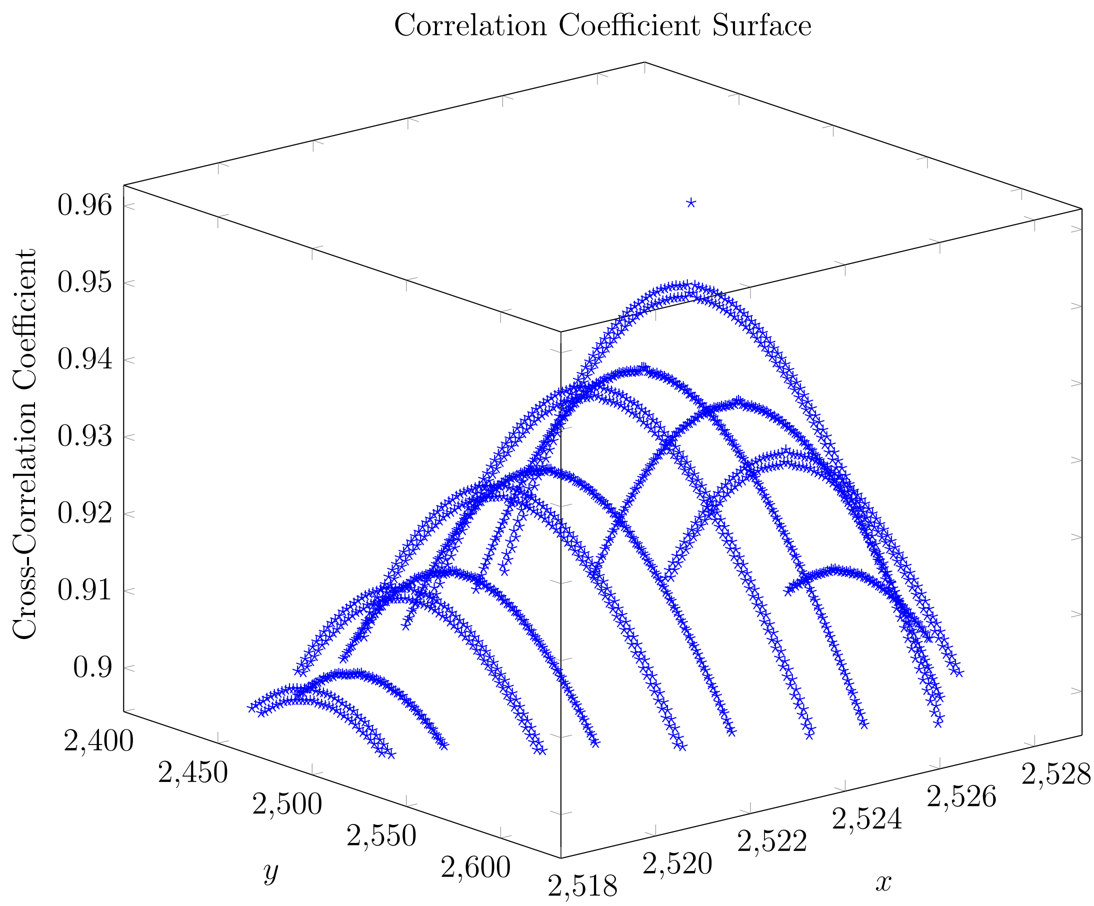



Figure 8.2: Filtered correlation coefficient data

The resulting surface is similar to that of a general quadratic surface.

$$R(x, y) = a_0x^2 + a_1x + a_2y^2 + a_3y + a_4 \tag{8.1}$$

The filtered data is then input into a surface fitting algorithm which utilizes a maximum likelihood estimator to fit the data to a quadratic surface.

Next $x_i$, $y_i$, and $R_i$(correlation coefficient) data is used to compute the $M$ and $\vec{c}$ matrices as seen in equations (8.2) and (8.3).

$$M = \begin{bmatrix} \sum x_i^2 & \sum x_i^3 & \sum x_i^2y_i^2 & \sum x_i^2y_i & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_iy_i^2 & \sum x_iy_i & \sum x_i \\ \sum y_i^2x_i^2 & \sum y_i^2x_i & \sum y_i^4 & \sum y_i^3 & \sum y_i^2 \\ \sum y_ix_i^2 & \sum y_ix_i & \sum y_i^3 & \sum y_i^2 & \sum y_i \\ \sum x_i^2 & \sum x_i & \sum y_i^2 & \sum y_i & n^2 \end{bmatrix} \tag{8.2}$$

$$\vec{c} = \begin{bmatrix} \sum R_ix_i^2 & \sum R_ix_i & \sum R_iy_i^2 & \sum R_iy_i & \sum R_i \end{bmatrix} \tag{8.3}$$

$$\vec{\alpha} = M^{-1}\vec{c} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \tag{8.4}$$

Multiplying the inverse of the $M$ matrix with the $\vec{c}$ yields coefficient matrix $\vec{\alpha}$. These coefficients are then used to populate the equation for a three dimensional quadratic surface fit.

The partial derivatives of the original quadratic equation are derived and set equal to zero in order to determine the point where the slope of the quadratic surface is zero.

$$\frac{\partial R}{\partial x} = 2a_0 x + a_1 = 0 \tag{8.5}$$

$$\frac{\partial R}{\partial y} = 2a_2 y + a_3 = 0 \tag{8.6}$$

The maximum x and y values of the estimated function.

$$x_{max} = \frac{-a_1}{2a_0} \tag{8.7}$$

$$y_{max} = \frac{-a_3}{2a_2} \tag{8.8}$$

Using this method both effectively averages the effect of noise in the image and allows for sub-pixel estimation of the most likely position of the features on the image [4]. This form also allows for a straight forward estimation of the position measurement uncertainty. The Jacobian is defined as.

$$A = \begin{bmatrix} \frac{\partial x}{\partial a_0} & \frac{\partial x}{\partial a_1} & \frac{\partial x}{\partial a_2} & \frac{\partial x}{\partial a_3} & \frac{\partial x}{\partial a_4} \\ \frac{\partial y}{\partial a_0} & \frac{\partial y}{\partial a_1} & \frac{\partial y}{\partial a_2} & \frac{\partial y}{\partial a_3} & \frac{\partial y}{\partial a_4} \end{bmatrix} \tag{8.9}$$

Once the jacobian $A$ matrix is created, the covariance matrix for $x_{max}$ and $y_{max}$, $V$ can then be generated.

$$V = AM^{-1}A^T \tag{8.10}$$

Uncertainty can be determined by taking the square root of the variance.

# CHAPTER 9: MEASUREMENT UNCERTAINTY

## 9.1 Camera Position Measurement Uncertainty

To measure the uncertainty of the correlation-based position measurement, 40 measurements were collected at one second intervals. The standard deviation of these measurements is 4.85 nm. The repeatability uncertainty does not however take into account the Abbe errors caused by changes in the mounting alignment. The camera system is mounted on a rotational hinge which is used to align the camera's horizontal axis to the wire sample's axis of translation. This mechanism and its adjustment screw are subject to thermal expansion. This thermal expansion influences the angular alignment between the axes. Changes in axis alignment lead to Abbe error. Additionally the wire and the wire holder both experience thermal expansion, which can contribute to uncertainty of the position measurement.

Table (9.1) lists the variables associated with the camera and sample mounting which are used in the following calculations of uncertainty.

Table 9.1: Variables used to determine DIC based location measurement uncertainty

| **specification** | **value** | **discription** |
|---|---|---|
| $r_i$ | 0.06 mm | length of wire in camera coordinates |
| $\Delta r$ | 0.009 nm | change in length of wire<br>in camera coordinates due to thermal expansion |
| $l_i$ | 76.2 mm | initial length of flexure lever arm |
| $\Delta l$ | 23.6 nm | change in length of flexure lever arm<br>due to thermal expansion |
| $h_i$ | 25.4 mm | initial height of flexure angle adjustment screw |
| $\Delta h$ | 158.75 nm | change in height of flexure angle adjustment screw<br>due to thermal expansion |
| $\Delta T$ | 0.5°C | change in temperature within environmental enclosure |
| $\alpha_{abs}$ | $31 \times 10^{-6}\ \frac{\mathrm{m}}{\mathrm{m} \cdot \mathrm{K}}$ | CTE of abs plastic |
| $\alpha_{steel}$ | $12.5 \times 10^{-6}\ \frac{\mathrm{m}}{\mathrm{m} \cdot \mathrm{K}}$ | CTE of steel |
| $\Delta_\theta$ | $2 \times 10^{-4\circ}$ | change in angle of flexure hinge |
| $\sigma l_i$ | 0.127 mm | uncertainty of initial length of flexure lever arm |
| $\sigma \Delta l$ | 1.18 µm | uncertainty of change in length of flexure lever arm<br>due to thermal expansion |
| $\sigma h_i$ | 0.0762 mm | uncertainty of initial height of flexure angle adjustment screw |
| $\sigma \alpha_{abs}$ | $3.1 \times 10^{-6}\ \frac{\mathrm{m}}{\mathrm{m} \cdot \mathrm{K}}$ | uncertainty of the CTE of ABS plastic |
| $\sigma \alpha_{steel}$ | $1.25 \times 10^{-6}\ \frac{\mathrm{m}}{\mathrm{m} \cdot \mathrm{K}}$ | uncertainty of the CTE of steel |
| $\sigma_\Delta \theta$ | $1 \times 10^{-5\circ}$ | uncertainty of flexure hinge angle |
| $\sigma_{\Delta T}$ | .01°C | uncertainty of the change in temperature |

The change in length of the ABS plastic lever and the adjustment screw due to change in temperature is.

$$\Delta l = \alpha_{abs} l_i \Delta T \tag{9.1}$$

$$\Delta h = \alpha_{steel} h_i \Delta T \tag{9.2}$$

The change of angle, $\Delta_\theta$, due to change in temperature.

$$\Delta_\theta = \theta_2 - \theta_1 \tag{9.3}$$

$$\theta_1 = tan^{-1}\left(\frac{h_i}{l_i}\right) \tag{9.4}$$

$$\theta_2 = tan^{-1}\left(\frac{h_i + \Delta h}{l_i + \Delta l}\right) \tag{9.5}$$

The uncertainty of $\Delta l$, $\Delta h$, and $\Delta r$ can be calculated using equations (9.6), (9.7), and (9.8)

$$\sigma_{\Delta l} = \sqrt{(\alpha_{abs}\Delta T\sigma l_i)^2 + (l_i\Delta T\sigma_{\alpha abs})^2 + (l_i\alpha_{abs}\sigma_{\Delta T})^2} \tag{9.6}$$

$$\sigma_{\Delta h} = \sqrt{(\alpha_{steel}\Delta T\sigma h_i)^2 + (h_i\Delta T\sigma_{\alpha steel})^2 + (h_i\alpha_{steel}\sigma_{\Delta T})^2} \tag{9.7}$$

$$\sigma_{\Delta r} = \sqrt{(\alpha_{steel}\Delta T\sigma r_i)^2 + (r_i\Delta T\sigma_{\alpha steel})^2 + (r_i\alpha_{steel}\sigma_{\Delta T})^2} \tag{9.8}$$

The uncertainty of the angle between the horizontal axis of the camera and the translation axis of the camera is:

$$\sigma_\theta = \sqrt{\left(\frac{\partial\theta}{\partial h_i}\sigma h_i\right)^2 + \left(\frac{\partial\theta}{\partial\Delta h}\sigma\Delta h\right)^2 + \left(\frac{\partial\theta}{\partial l_i}\sigma l_i\right)^2 + \left(\frac{\partial\theta}{\partial\Delta l}\sigma\Delta l\right)^2} \tag{9.9}$$

Where the sensitivities to each component are:

$$\frac{\partial\theta}{\partial h_i} = \frac{l_i + \Delta l}{\Delta h^2 + 2\Delta h h_i + h_i^2 + (l_i + \Delta l)^2} + \frac{h_i}{l_i^2 + h_i^2} \tag{9.10}$$

$$\frac{\partial\theta}{\partial\Delta h} = \frac{l_i + \Delta l}{\Delta h^2 + 2\Delta h h_i + h_i^2 + (l_i + \Delta l)^2} \tag{9.11}$$

$$\frac{\partial\theta}{\partial l_i} = \frac{h_i + \Delta h}{(l_i + \Delta l)^2\left(\frac{(h_i+\Delta h)^2}{(l_i+\Delta l)^2} + 1\right)} + \frac{h_i}{h_i^2 + l_i^2} \tag{9.12}$$

$$\frac{\partial\theta}{\partial \Delta l} = \frac{h_i + \Delta h}{(l_i + \Delta l)^2\left(\frac{(h_i+\Delta h)^2}{(l_i+\Delta l)^2} + 1\right)} \tag{9.13}$$

The uncertainty of the $x$ position, as measured by the camera:

$$\sigma_x = \sqrt{(cos(\Delta\theta)\sigma_{ri})^2 + (cos(\Delta\theta)\sigma\Delta\theta)^2 + ((-r_i sin(\Delta\theta) - \Delta r sin(\Delta\theta))\sigma\theta)^2} \tag{9.14}$$

The estimated measurement uncertainty of the $x$ position measurement, including Abbe error, is $\pm$ 105 nm. The Abbe error of the $x$ position measurement is very sensitive to the uncertainty of the temperature measurement of the system. If the temperature of the system could be controlled to $\pm$ 0.1°C then this uncertainty could be reduced to $\pm$ 24 nm.

## 9.2 Encoder Position Measurement Uncertainty

The encoder axial alignment system is configured in a similar manner to the camera system, and as such, the uncertainty equations are also very similar. The largest difference between the two mounting configurations is that the camera system is mounted on an ABS plastic flexure hinge and the encoder is mounted on a steel flexure hinge. To evaluate the repeatability uncertainty of the encoder position measurement, 40 measurements were taken at a time interval of 1 second. The standard deviation of these measurements is 3.5 nm. The measurement uncertainty in the current environment is estimated to be ± 48 nm, largely due to Abbe error. The all steel construction of this structure significantly reduces the effect of Abbe error. If the temperature could be controlled to ± 0.1 °C, then the estimated uncertainty reduces to 10.2 nm. Analysis of the uncertainty of the encoder can be found in Appendix C.

## 9.3 Static Drift

To measure the thermal drift of the instrument, the instrument is allowed to make continuous measurements for an 8 hour period. During this time, the maximum drift of the instrument is approximately 800 nm. A temperature sensor is placed within the environmental enclosure to allow for the measurement and compensation of thermal effects. The instrument's drift shows a cyclic behavior with approximately the same frequency as the temperature in the enclosure changes. In order to compensate for the effects of temperature, the temperature is scaled and then subtracted from the measured position. The figure below shows that the uncorrected camera position measurement has a mean value of $-322$ nm and a standard deviation of 201 nm. When the position is corrected for temperature variation, the standard deviation becomes 120 nm. The instrument has a lumped temperature sensitivity of approximately 1900 $\frac{\text{nm}}{°\text{C}}$. The environmental temperature, which the instrument is tested in, is only controlled to 0.5°C which allows for approximately $\pm$ 465 nm of drift. If the temperature could be controlled to 0.1°C then it should be possible to reduce the uncorrected thermal drift to $\pm$ 195 nm.

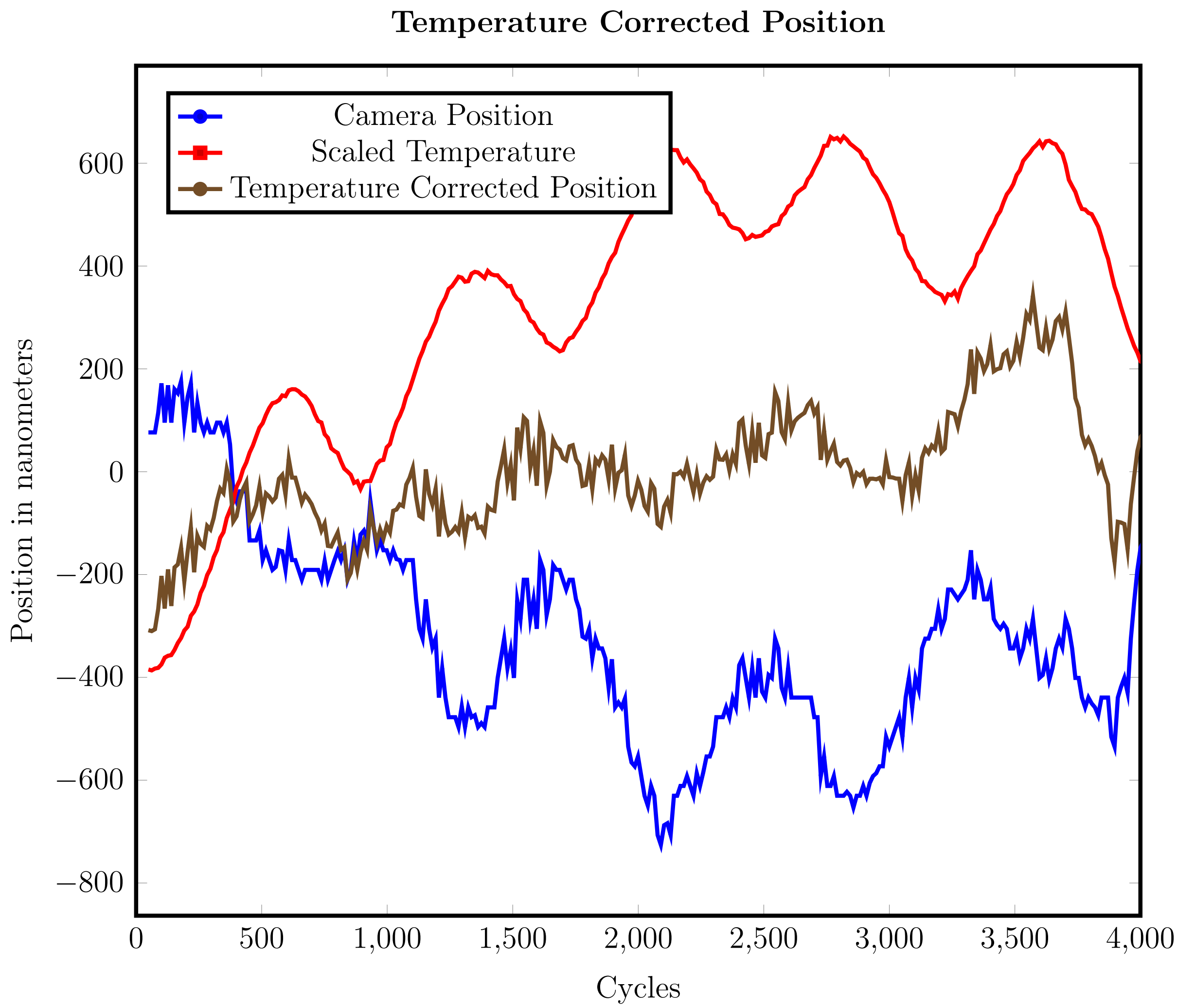



Figure 9.1: Temperature corrected camera position

### 9.4 Dynamic Drift

In order to analyze how well the position measurement generated using DIC compares with the encoder position, the linear stage is actuated at varying rates in both directions. The camera and encoder position measurements, as well as the calculated error between them, are shown in figure (9.2). The average error is 4.2 nm, and the standard deviation is 25.8 nm.

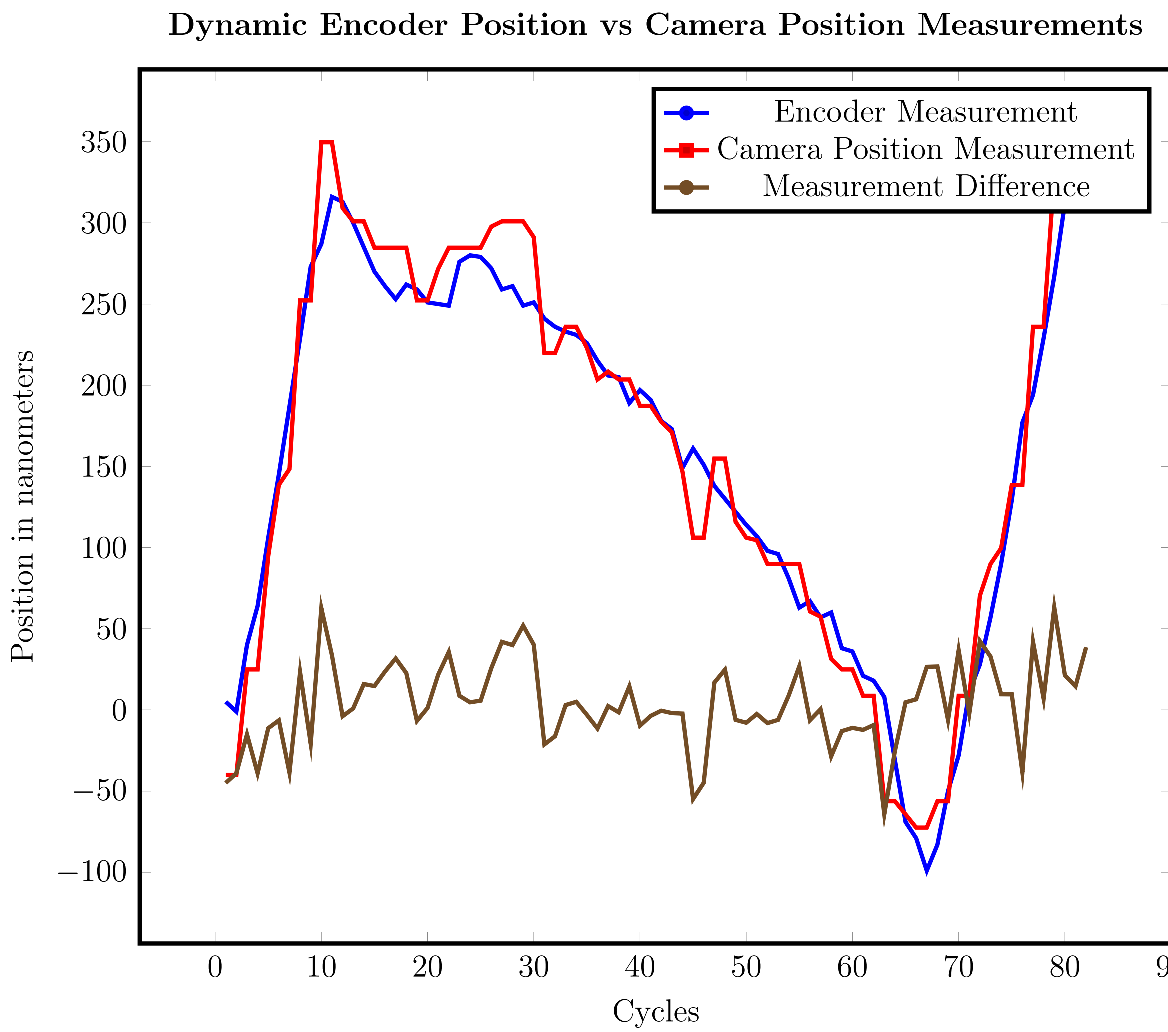



Figure 9.2: Encoder vs camera

### 9.5 Combined Camera and Encoder Measurement Uncertainty

The encoder produces a continuous global position measurement, while the camera produces a local discontinuous position measurement. In order to generate a final position measurement, the camera measurement and the encoder measurement must be summed. The camera position measurement for the target features is only tracked and recorded when the features are in view of the camera; any thermal drift that occurs when the features are out of view results in measurement error. Thus the instrument cannot achieve an uncertainty less than the temperature corrected drift within the system which is around $\pm$ 100 nm in the current environment. This uncertainty could be significantly reduced if the environmental temperature were more strictly controlled.

# CHAPTER 10: CONCLUSIONS

Individually, the encoder and camera position measurements achieve the requisite resolution to fulfill the design goals of this project; however, the most significant limitation is thermal drift. This leads to the conclusion that this technique is viable for use in a high dynamic range creep measurement system, which is optimized to reduce thermal effects. One of the advantages of this technique compared to similar techniques, is the ability to achieve the desired performance without the need to "mark" samples before measuring them. Another advantage is that this system can be constructed at a much lower cost than other comparable systems which utilize scanning electron microscope imagery or other similarly expensive technologies to achieve comparable results. A third advantage of this technique is its ability to be expanded to observe multiple experiments within the same test setup.

When the optical magnification system for this instrument was designed, it was not known to what extent sub-pixel resolution could be achieved. Therefore, the system was designed with the assumption that a factor of two increase in resolution could be achieved using sub-pixel resolution algorithms. Subsequently the magnification for this system was designed around this requirement. After completing this proof of concept, it is estimated that sub-pixel algorithms can provide at least an order of magnitude improvement in effective resolution. With this in mind the magnification requirement of the system could likely be reduced and still achieve comparable results.

# CHAPTER 11: FUTURE WORK

## 11.1 Thermal Control

The largest source of error in the system, which limits the system's performance, appears to be a result of thermal effects, from the temperature fluctuation in the room that this instrument has been constructed in. By building active temperature control into the next iteration of this experiment, it may be possible to obtain a significant reduction in uncertainty. For the purpose of expedience large portions of the imaging assembly were constructed from ABS plastic, which has a relatively high coefficient of thermal expansion compared to other potential material choices which could be employed in future systems.

## 11.2 Creep Measurement Instrument Design

The purpose of this design is to investigate the potential uncertainties associated with obtaining a creep measurement using a scanning DIC-based instrument. Ultimately the intent is that a full-scale instrument would be created using the knowledge gained through this project. Having proved that DIC based position measurement is a viable technique for tracking features on a wire, a fully functional creep measurement instrument, based on this principle could be constructed. In order to do this, the length scale of the system would need to be elongated from tens of millimeters to hundreds of millimeters. In the current testing configuration, the wire sample has been positioned horizontally with only enough load placed on it to keep the wire relatively taut. In the next iteration of this instrument, the wire would need to be loaded in a mechanism which keeps a constant tension on the wire sample even as the wire experiences creep. The most straight forward method for doing this would be to suspend a known weight from a vertical wire, which is fixed in position at the top. Subsequently, it is also desirable that the system be able to scan across multiple wires in order to complete multiple tests simultaneously. This could be achieved by fixturing multiple parallel wires, and increasing the instrument's axis of travel, in the direction perpendicular to the wire, so that the camera can traverse between all of the wires.

# REFERENCES


[1] F. Di Gioacchino and J. Quinta da Fonseca, "Plastic strain mapping with sub-micron resolution using digital image correlation," *Experimental Mechanics*, vol. 53, pp. 743–754, Jun 2013.

[2] M. J. Telfer, M. A. Coe, B. A. Cottom, and J. R. Price, "Apparatus and method for optically measuring creep," Aug. 26 2014. US Patent 8,818,078.

[3] A. Sousa, J. Xavier, M. Vaz, J. Morais, and V. Filipe, "Cross-correlation and differential technique combination to determine displacement fields," *Strain*, vol. 47, pp. 87–98, 2011.

[4] M. Debella-Gilo and A. Kääb, "Sub-pixel precision image matching for measuring surface displacements on mass movements using normalized cross-correlation," *Remote Sensing of Environment*, vol. 115, no. 1, pp. 130–142, 2011.

[5] K. G. Harding and Y. Liao, "Method and system for creep measurement," Dec. 8 2015. US Patent 9,207,154.

[6] E. Madhi and P. B. Nagy, "Sensitivity analysis of a directional potential drop sensor for creep monitoring," *NDT E International*, vol. 44, no. 8, pp. 708 – 717, 2011.

[7] K. Naumenko and H. Altenbach, *Modeling of creep for structural analysis*. Springer Science & Business Media, 2007.

[8] K. Lawton, M. and S. Patterson, R., *Long-term Creep Measurements of 302 Stainless Steel and Elgiloy*. American Society for Precision Engineering:, 2018.

[9] J. M. Geary, *Introduction to lens design: with practical ZEMAX examples*. Willmann-Bell Richmond, VA, USA:, 2002.

[10] A. V. Arecchi, R. J. Koshel, and T. Messadi, "Field guide to illumination," SPIE, 2007.

[11] E. C. Fest and S. of Photo-optical Instrumentation Engineers, *Stray light analysis and control*. SPIE press Bellingham, 2013.

[12] S. T. Smith, *Flexures: elements of elastic mechanisms*. Crc Press, 2014.

[13] R. C. Gonzalez, "Digital image processing," 2018.

# APPENDIX A: MATLAB CODE

## A.1 blur vs noise simulation

```
clc
clear

format long;

img = imread('wire_sample2.jpg');
img_gray=rgb2gray(img);

noiselevel(img_gray)
noisem=.0005;
variance=.0015;
sigma=1;
scale=.001
rect =[1500,900,1000,2000];

%loading and manipulating images--------------------------------
img_gray=rgb2gray(img);

img_gray_blur=imgaussfilt(img_gray,sigma);
img_gray_noise_and_blur=imnoise(img_gray_blur,'gaussian',noisem,variance);
reps=10
for sigma=1:reps;
    sigmamult=sigma*scale
```

```
    for signoise=1:reps;
        noisevar=(signoise^2)*scale

%cropped images————————————————————
crop_sample = imcrop(img_gray,rect);
img_crop_with_blur= imgaussfilt(crop_sample,sigma);
img_crop_with_noise_and_blur=imnoise(img_crop_with_blur,'
    gaussian',noisem,noisevar);

 cor3=normxcorr2(img_crop_with_noise_and_blur,
     img_gray_noise_and_blur);

%plotting ——————————————————————————
figure(1)
subplot(3,3,1)
 imshow(img_gray);
 title('grayscale wire')

 subplot(3,3,2)
 imshow (crop_sample);
 title('cropped image')

 subplot(3,3,4);
 imshow(img_gray_blur);
 title('grayscale wire')

 subplot(3,3,5);
```

```
imshow(img_crop_with_blur);
 title('cropped image with blur')

subplot(3,3,7);
imshow (img_gray_noise_and_blur);
 title('wire with noise and blur')

subplot(3,3,8)
imshow (img_crop_with_noise_and_blur);
 title('crop with noise and blur')
pause(1)

cc_coef_3(sigma,signoise)=max(max(cor3))
[ypeak3, xpeak3] = find(cor3==max(cor3(:)))


    end
end
figure(2);
 surf(cc_coef_3);

X=[1:signoise]
Y=[1:sigma]
 xscale=X*scale
 yscale=Y*scale
 [y,x]=meshgrid(yscale,xscale)

```

```
x_vec=reshape(x,sigma.*signoise,1)
y_vec=reshape(y,sigma.*signoise,1)
cc_coef_3_vec=reshape(cc_coef_3,sigma.*signoise,1)
figure(2)
scatter3(x_vec(:,1),y_vec(:,1),cc_coef_3_vec(:,1))
xlabel=('noise')
ylabel=('blur')

cubic_fit=fit([x_vec,y_vec],cc_coef_3_vec,'poly11')
[fx, fy] = differentiate(cubic_fit, [x_vec, y_vec])

figure(3)
plot(cubic_fit)

data= [x_vec y_vec cc_coef_3_vec]

csvwrite('cross_cor.csv',data)
```

## A.2 phase lag corrected data analysis

```
clc;
clear;
M = csvread('data11.csv',170,1);
shift= 61;
axis([0 10 0 10 0 2000]);
sizem=size(M,1);
itr=[1:sizem]';
data=[itr,M];
itr=data(:,1);
enc=data(:,2);
cam=data(:,3);
cam_zero=(cam-cam(1));
amp=1744;
temp=data(:,6);
temp2=(temp-temp(1));
temp2_amp=temp2*amp;
enc_shift=enc(1:end-shift);
cam_zero_shift=cam_zero(shift:end);
temp2_shift=temp2_amp(1:end-shift+1)-156;
itr_shift=itr(shift:end);
cam_comp_shift=cam_zero_shift+temp2_shift;

figure(1)
plot(itr_shift,cam_zero_shift,itr_shift,temp2_shift,itr_shift
    ,cam_comp_shift,'linewidth',4)
```

```

title('shifted')

meanval_unc=mean(cam_zero_shift)
stddev_unc=std(cam_zero_shift)

meanval=mean(cam_comp_shift)
stddev=std(cam_comp_shift)

data1 = cat(1,[itr_shift ,cam_zero_shift,temp2_shift,
    cam_comp_shift]);
csvwrite('themalerror.csv',data1);
```

APPENDIX B: C++ CODE

```
#include <iostream>
#include <eib7.h>
#include <chrono>
#include <ctime>
#include <unistd.h>
#include <cstdio>
#include <signal.h>
#include <fstream>
#include <thread>
#include <stddef.h>
#include "opencv2/opencv.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/objdetect/objdetect.hpp"
#include <opencv2/imgcodecs.hpp>
#include <opencv2/imgcodecs/imgcodecs.hpp>
#include "opencv2/highgui/highgui.hpp"
#include <eigen3/Eigen/Core>
#include <opencv2/core/eigen.hpp>
#include <ueye.h>
#include <time.h>
#include <LabJackM.h>
#include "LJM_Utilities.h"
#include <mutex>
#include <condition_variable>
```

```
#include <cmath>
#include <vector>
#include <algorithm>
#include <Eigen>


#define EIB_TCP_TIMEOUT        5000    /* timeout for TCP connection in ms       */
#define NUM_OF_AXIS            4       /* number of axes of the EIB              */
#define MAX_TEXT_LEN           200     /* maximum size of console input string */


HIDS hCam = 0;

using namespace std;
using namespace cv;
using namespace Eigen;


char quit;
int quitnum=0;
bool ready = false;
mutex m;
mutex m2;
mutex m3;
```

```
mutex m4;
mutex m5;
condition_variable cond_var;

double loc=0;
double loc2=0;
double locy=0;
double locy2=0;

double zero;
long int encpos;
long int pos = encpos;
int handle;
double temperature;
double avg2;

double avgfitx=0;double fitx1=0;double fitx2=0;


void CheckError(EIB7_ERR error);
void  PollPosition(EIB7_AXIS axis, int enc_type, long int& a)
    ;
int picomotor(int handle,int n,int t, int Dist,double Spd,
    double dir);
int picomotor2(int handle, int n,int t, int Dist,double Spd,
    double diry);
void vertcontrol(int handle);
```

```

void encavg();
void encoder();
Mat camera();
int temp( int handle);
void vertcontrol(int handle);
void data();


class coor_row
{
    public:
        long double x,y,z;
 coor_row(long double X,long double Y,long double Z)
    {
        x=X;
        y=Y;
        z=Z;
    }
    friend ostream& operator<<(ostream&os, const coor_row& xyz);
    friend ofstream& operator<<(ofstream&fs, const coor_row& xyz);
};

//overload the << operator to display the result of my row
ostream& operator<<(ostream& os,const coor_row& xyz)
```

```
{
    os<< xyz.x << " " << xyz.y << " " << xyz.z;
    return os;
}
ofstream& operator<<(ofstream& fs,const coor_row& xyz)
{
    fs<< xyz.x << " " << xyz.y << " " << xyz.z;
    return fs;
}

int main()
{
 int handle;
handle = OpenOrDie(LJM_dtANY, LJM_ctANY, "LJM_idANY");

thread encoderaverage(encavg);
thread encoderthread (encoder);
thread camthread (camera);
thread tempthread(temp, handle);
thread vertcontrolthread (vertcontrol, handle);
thread datarec (data);

//--------------------------------------------------------------------------

#define POS_SPEC_INCR "Status word: 0x%04X Position (int):
    %010lld = %11.4f signal periods "

```

```
#define POS_SPEC_INCR  "Status word: 0x%04X Position (int): %010lld  = %11.4f signal periods "

using namespace std;

 int exit = 3;

        // initialize vairiables
            //int err;
            //int handle;
            int n=0;
            int t;
            int Dist;
            double Spd;
            char D;
            double dir; //output state (0 = reverse, 1= forward)
        long int pos;
        //double zero;
        int target;
        double delta=1000000000000000000;
        int endcond= 10;
        double rep=3;
        double i;
        double j;
        double tot;
        double avg;
        double encrep;
```

```
      double posnew=0;
      double posavg;


      sleep(3);



     m.lock();
      pos;
      avg2;
      cout<< ""<< endl;
      cout << "this is the initial possition " << encpos <<endl
         ;
      zero= avg2;
      cout<< "this is what it believes the zero offset should
         be "<< zero << endl;cout <<endl;

     m.unlock();

while (exit != 0)
{  tot=0;
double locnano=loc;
  //  sleep(7);

      for (i = 1; i <= rep; ++i){
      locnano=loc;
      tot=tot+locnano;

```

```
    }
    avg= tot/rep;

    cout<< "would you like to quit? press Q to quit or any other key to continue"  << endl;
    cin >> quit;


    if (quit != 'q')
    {quitnum =0;}

    else

    if (quit == 'q')
    {quitnum=1;
    ready = true;}

  // Determines what the target destination for the stage is.
        cout << "what is the destination location: ";
        cin >> target;
    delta=1000000000000000000;

while (delta > endcond or delta < -endcond)
{
    // finds the position of the stage from the encoder at the beginning of the loop and displays it.

```

```
m2.lock();
      encpos;
      avg2;

      cout<< ""<< endl;

      double a=avg2;
      cout << "the position is: "<< avg2 << endl;

      delta = target -a ;
      cout << "delta is: " << delta << endl;

      cout << "zero is: " << zero << endl;

            if (delta > 0)
            {
                  dir = 0;
                  Spd=99999999;
                  t=1;
      }
      else
      if (delta < 0)
      {
            dir=1;
            Spd=99999999;
            t=1;
```

```
    }
    else
    {
    cout << "This is not a valid direction " << endl;
    }
    cout << "the direction is " << dir << endl;


        t=Dist=1;




    picomotor(handle,n,t,Dist,Spd,dir);

        double b=-avg2+zero;


        double c = b-a;

        m2.unlock();


        cout << "------------------------------------------------" <<
            endl;
}

```

```
}

    encoderaverage.join();
    encoderthread.join();
    camthread.join();
    tempthread.join();
    vertcontrolthread.join();
    datarec.join();

    return 0;
}



Mat camera()
{

Mat sample;

Mat sample7;
Mat resized;

sample = imread("sample15.jpg",CV_LOAD_IMAGE_GRAYSCALE);

sample7 = imread("sample16.jpg",CV_LOAD_IMAGE_GRAYSCALE);


```

```
imshow("sample", sample);
imshow("sample7", sample7);

 Mat debug_img;

Mat result_mat;
Mat result_mat2;
MatrixXf img_eig;


//

 MatrixXd M(5,5);
    Eigen::Matrix<double,5,1> C = Eigen::Matrix< double
        ,5,1 >::Zero();
    //MatrixXd minv;
    Eigen::Matrix<double,5,5> minv = Eigen::Matrix<double
        ,5,5 >::Zero();
    Eigen::Matrix<double,5,1> A = Eigen::Matrix<double,5,1 >::
        Zero();



//
```

```



HIDS hCam = 0;
char* pMem = NULL;
int memID = 0;
// Initialize the camera. The second input is for windows and
      by passing NULL you tell it to run in bit map mode
int nRet = is_InitCamera(&hCam,NULL);
    if (nRet == IS_SUCCESS)
    {cout << "the camera was initialized"<< endl;}
    else
    {cout<< "camera not initialized"<< endl;}



    // This command Determines how many Ueye cameras are
        connected to the computer and returns a value "num"
    int num = 0;
    nRet=is_GetNumberOfCameras(&num);
      cout << "number of connected cameras: "<< num<< endl;

// This function sets the pixel clock for the camera.
     UINT nPixelClockDefault = 10; //this was set at 21
     nRet = is_PixelClock(hCam, IS_PIXELCLOCK_CMD_SET, (void*)
         &nPixelClockDefault, sizeof(nPixelClockDefault));
     if (nRet == IS_SUCCESS)
```

```
  {cout << "the pixel clock was set"<< endl;}
  else
  {cout<< "pixel clock was not set"<< endl;}



 // This function sets the exposure of the camera.
  double exposure=900;
  nRet = is_Exposure(hCam, IS_EXPOSURE_CMD_SET_EXPOSURE, &exposure, sizeof(exposure));
  printf("exposure stat %d set exposure %f \n",nRet,exposure);

  // Set the color mode of the camera
   INT colorMode = IS_CM_SENSOR_RAW8;
   nRet = is_SetColorMode(hCam,colorMode);

   if (nRet == IS_SUCCESS){
       cout << "Camera color mode succesfully set!" << endl;
   }




// this section of the querys the camera for a list of supported image formats and the associated parameters for each
```

```
UINT count;

UINT bytesNeeded = sizeof(IMAGE_FORMAT_LIST);

nRet = is_ImageFormat(hCam, IMGFRMT_CMD_GET_NUM_ENTRIES, &
    count, sizeof(count));

bytesNeeded += (count - 1) * sizeof(IMAGE_FORMAT_INFO);

void* ptr = malloc(bytesNeeded);



// Create and fill list

IMAGE_FORMAT_LIST* pformatList = (IMAGE_FORMAT_LIST*) ptr;

pformatList->nSizeOfListEntry = sizeof(IMAGE_FORMAT_INFO);

pformatList->nNumListElements = count;

nRet = is_ImageFormat(hCam, IMGFRMT_CMD_GET_LIST, pformatList
    , bytesNeeded);

IMAGE_FORMAT_INFO formatInfo;

```

```
// This code creates integer values which indicate the height and width of the image in pixels.
// Changing the index of "FormatInfo" changes the image format
formatInfo = pformatList->FormatInfo[1];
    int width = formatInfo.nWidth;
    cout << "width: " << width << endl;
    int height = formatInfo.nHeight;
    cout << "height: " << height << endl ;
// initializes an OpenCV matrix "mat" with the Height and Width of the selectec image format.
 Mat mat (height, width, CV_8UC1);
nRet = is_AllocImageMem(hCam, width, height, 8, &pMem, &memID);
  if (nRet == IS_SUCCESS)
   {cout << "the memory was allocated"<< endl;}
   else
```

```
    {cout<< "memory was not allocated"<< endl;}
    //------------------------------------------------------------------------------

nRet = is_SetImageMem(hCam, pMem, memID);
     if (nRet == IS_SUCCESS)
    {cout << "the memory was activated"<< endl;}
    else
    {cout<< "memory was not activated"<< endl;}


     nRet = is_ImageFormat(hCam, IMGFRMT_CMD_SET_FORMAT, &formatInfo.nFormatID, sizeof(formatInfo.nFormatID));


while(true){

 //initializing the variables to generate the Matrix
     long double sumx=0; long double sumy=0; long double sumxx=0; long double sumyy=0; long double sumxy=0; long double sumyx=0; long double sumxxx=0; long double sumxxxx=0; long double sumyyy=0;
     long double sumyyyy=0; long double sumxxyy=0; long double sumxxy=0; long double sumxyy=0; long double sumyyxx=0; long double sumyxx=0; long double sumyyx=0;

```

```
    //iniitializing the variables required to generate the C matrix
    long double sumzxx=0; long double sumzx=0; long double sumzyy=0; long double sumzy=0; long double sumz=0;

    int DELAY_CAPTION = 1500;
    int DELAY_BLUR = 100;
    int MAX_KERNEL_LENGTH = 31;

   is_FreezeVideo(hCam, IS_WAIT);
   char* pMem_b;
         int retInt = is_GetActiveImageMem(hCam, &pMem,&memID);

   memcpy(mat.ptr(), pMem, mat.cols * mat.rows);
   Mat amp =(mat*3)-55;
   Mat thresh;
   Mat resized_thresh;
   Mat resized_result;
   double thresh_val=.3;
   int counter=0;



  cvtColor(amp, debug_img, CV_GRAY2BGR);


```

```
for ( int i = 1; i < MAX_KERNEL_LENGTH; i = i + 2 )
        { GaussianBlur( amp, amp, Size( i, i ), 0, 0 );}


//--first image cross correlation
    ---------------------------------------------------------------


 int match_method = CV_TM_CCORR_NORMED;
                    matchTemplate(amp, sample, result_mat,
                        match_method);

                    double minVal; double maxVal;
                    Point minLoc, maxLoc, matchLoc;
          minMaxLoc(result_mat, &minVal, &maxVal, &minLoc, &
              maxLoc, Mat() );
     matchLoc = maxLoc;



     double mincoeff=.70;

     if (maxVal>= mincoeff)

     {
                    rectangle(
                    debug_img,
                    matchLoc,
```

```
                Point(matchLoc.x + sample.cols , matchLoc.y +
                    sample.rows),
                CV_RGB(255,0,0),
        3);

        loc = matchLoc.x;
        locy = matchLoc.y;
        Rect roi;
        roi.width = 100;
        roi.height = 100;
        roi.x = matchLoc.x-roi.width/2;
        roi.y = matchLoc.y-roi.height/2;


        Mat crop = result_mat(roi);
        //imshow("crop", crop);
        cv2eigen(crop,img_eig );

        int sizex=img_eig.rows();
        int sizey=img_eig.cols();


        long double z[sizey][sizex];
        long double x[sizex], y[sizey];
        vector<coor_row>new_row;

        for (int j=0; j< sizex; j++)
```

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

            {x[i]=j;
            y[j]=i;
            z[i][j]=img_eig(i,j);

            coor_row xyz(x[i],y[j],z[i][j]);
if (z[i][j]>=thresh_val)
{
            new_row.push_back(xyz);
}

else {counter++;}




}
}

cout << endl;
for (int i=0; i<(sizex*sizey-counter); i++){

              //the m matrix
              sumx+=new_row[i].x;
              sumy+=new_row[i].y;
```

```
        sumxx+=new_row[i].x*new_row[i].x;
        sumxxx+=new_row[i].x*new_row[i].x*new_row[i].x;
        sumxxxx+=new_row[i].x*new_row[i].x*new_row[i].x*
            new_row[i].x;
        sumyy+=new_row[i].y*new_row[i].y;
        sumyyy+=new_row[i].y*new_row[i].y*new_row[i].y;
        sumyyyy+=new_row[i].y*new_row[i].y*new_row[i].y*
            new_row[i].y;
        sumxy+=new_row[i].x*new_row[i].y;
        sumxyy+=new_row[i].x*new_row[i].y*new_row[i].y;
        sumxxy+=new_row[i].x*new_row[i].x*new_row[i].y;
        sumyx+=new_row[i].y*new_row[i].x;
        sumyyxx+=new_row[i].y*new_row[i].y*new_row[i].x*
            new_row[i].x;
        sumyyx+=new_row[i].y*new_row[i].y*new_row[i].x;
        sumyxx+=new_row[i].y*new_row[i].x*new_row[i].x;
        sumxxyy+=new_row[i].x*new_row[i].x*new_row[i].y*
            new_row[i].y;
        //the c matrix
        sumzxx+=new_row[i].z*new_row[i].x*new_row[i].x;
        sumzx+=new_row[i].z*new_row[i].x;
        sumzyy+=new_row[i].z*new_row[i].y*new_row[i].y;
        sumzy+=new_row[i].z*new_row[i].y;
        sumz+=new_row[i].z;
        }
```

```
            M << sumxxxx,sumxxx,sumxxyy,sumxxy,sumxx,
             sumxxx,sumxx,sumxyy,sumxy,sumx,
             sumyyxx,sumyyx,sumyyyy,sumyyy,sumyy,
             sumyxx,sumyx,sumyyy,sumyy,sumy,
             sumxx,sumx,sumyy,sumy,sizex*sizey;
             //cout << endl<< M<< endl;


             C << sumzxx, sumzx, sumzyy, sumzy, sumz;
             minv=M.inverse();
             A=M.inverse()*C;
             fitx1= matchLoc.x+C(1,0)/(-2*C(0,0));
             }

         //--second image cross correlation
           ----------------------------------------------------------------------


     counter=0;
             matchTemplate(amp, sample7, result_mat2,
                match_method);

     double minVal2; double maxVal2;
     Point minLoc2, maxLoc2, matchLoc2;
         minMaxLoc(result_mat2, &minVal2, &maxVal2, &
            minLoc2, &maxLoc2, Mat() );
     matchLoc2 = maxLoc2;
```

```
        //initializing the variables to generate the Matrix
       sumx=0;  sumy=0;  sumxx=0;  sumyy=0;  sumxy=0;  sumyx=0;
           sumxxx=0; sumxxxx=0; sumyyy=0;
       sumyyyy=0;  sumxxyy=0;  sumxxy=0;  sumxyy=0;  sumyyxx=0;
            sumyxx=0;  sumyyx=0;

      //iniitializing the variables required to generate the C
          matrix
       sumzxx=0;  sumzx=0;  sumzyy=0; sumzy=0;  sumz=0;

         if (maxVal2>= mincoeff)

          {
                     rectangle(
                     debug_img,
                     matchLoc2,
                     Point(matchLoc2.x + sample7.cols , matchLoc2.
                        y + sample7.rows),
                     CV_RGB(255,0,0),
           3);

           loc2 = matchLoc2.x;
           locy2 = matchLoc2.y;
           Rect roi2;
           roi2.width = 100;
           roi2.height = 100;
```

```
roi2.x = matchLoc2.x-roi2.width/2;
roi2.y = matchLoc2.y-roi2.height/2;


Mat crop2 = result_mat(roi2);

cv2eigen(crop2,img_eig );
int sizex=img_eig.rows();
int sizey=img_eig.cols();

long double z[sizey][sizex];
long double x[sizex], y[sizey];
vector<coor_row>new_row;

for (int j=0; j< sizex; j++)
{
for ( int i=0; i<sizey ;i++)

{x[i]=j;
y[j]=i;
z[i][j]=img_eig(i,j);

coor_row xyz2(x[i],y[j],z[i][j]);

if (z[i][j]>=thresh_val)
{
new_row.push_back(xyz2);
```

```
            }

            else {counter++;}

}
}

 //initializing the variables to generate the Matrix

cout << endl;
for (int i=0; i<(sizex*sizey)-counter; i++){

               //the m matrix
               sumx+=new_row[i].x;
               sumy+=new_row[i].y;
               sumxx+=new_row[i].x*new_row[i].x;
               sumxxx+=new_row[i].x*new_row[i].x*new_row[i].x;
               sumxxxx+=new_row[i].x*new_row[i].x*new_row[i].x*
                   new_row[i].x;
               sumyy+=new_row[i].y*new_row[i].y;
               sumyyy+=new_row[i].y*new_row[i].y*new_row[i].y;
               sumyyyy+=new_row[i].y*new_row[i].y*new_row[i].y*
                   new_row[i].y;
               sumxy+=new_row[i].x*new_row[i].y;
               sumxyy+=new_row[i].x*new_row[i].y*new_row[i].y;
               sumxxy+=new_row[i].x*new_row[i].x*new_row[i].y;
               sumyx+=new_row[i].y*new_row[i].x;
```

```
            sumyyxx+=new_row[i].y*new_row[i].y*new_row[i].x*
                new_row[i].x;
            sumyyx+=new_row[i].y*new_row[i].y*new_row[i].x;
            sumyxx+=new_row[i].y*new_row[i].x*new_row[i].x;
            sumxxyy+=new_row[i].x*new_row[i].x*new_row[i].y*
                new_row[i].y;
            //the c matrix
            sumzxx+=new_row[i].z*new_row[i].x*new_row[i].x;
            sumzx+=new_row[i].z*new_row[i].x;
            sumzyy+=new_row[i].z*new_row[i].y*new_row[i].y;
            sumzy+=new_row[i].z*new_row[i].y;
            sumz+=new_row[i].z;
            }


              M << sumxxxx,sumxxx,sumxxyy,sumxxy,sumxx,
              sumxxx,sumxx,sumxyy,sumxy,sumx,
              sumyyxx,sumyyx,sumyyyy,sumyyy,sumyy,
              sumyxx,sumyx,sumyyy,sumyy,sumy,
              sumxx,sumx,sumyy,sumy,sizex*sizey;



              C << sumzxx, sumzx, sumzyy, sumzy, sumz;


              minv=M.inverse();
```

```



                    A=M.inverse()*C;


double fitx3;

                    fitx2= matchLoc2.x+C(1,0)/(-2*C(0,0));
                    fitx3= C(1,0)/(-2*C(0,0));
            }

            avgfitx=(fitx1+fitx2)/2;

         resize(debug_img, resized_thresh,Size(), 0.25, 0.25);

            imshow("thresh", resized_thresh);

             // Check if we need to stop processing
         if ( (int)waitKey(10) >= 0 ){

                waitKey(1);}

               if (quitnum == 1)
                {break;}
}

   cout << "" << endl;
```

```
is_FreeImageMem(hCam,pMem,memID);
 if (nRet == IS_SUCCESS)
  {cout << "memory was cleared"<< endl;}
  else
 {cout<< "memory not cleared"<< endl;}

 nRet = is_ExitCamera(hCam);
    if (nRet == IS_SUCCESS)
  {cout << "camera was exited"<< endl;}
  else
{cout<< "camera not exited"<< endl;}


 destroyAllWindows();

 sleep(1);

}



void CheckError(EIB7_ERR error)
{
   if(error != EIB7_NoError)
   {
      char mnemonic[32];
      char message[256];
```

```

        EIB7GetErrorInfo(error, mnemonic, 32, message, 256);

        fprintf(stderr, "\nError %08X (%s): %s\n", error, mnemonic, message);
        exit(0);


    }
}


void  PollPosition(EIB7_AXIS axis, int enc_type, long int& b)
{



        unsigned short status;              /* status word */
        ENCODER_POSITION pos;               /* position value (integer) */
        double pos_sp;                      /* position value (signal periods) */

        /* read position from EIB */
        CheckError(EIB7GetPosition(axis, &status, &pos));

```

```
            CheckError(EIB7IncrPosToDouble(pos,&pos_sp));

             encpos = pos;

}

void encoder()
{
   EIB7_HANDLE eib;                  /* EIB handle
         */
   unsigned long ip;                 /* IP address of EIB
         */
   unsigned long num;                /* number of encoder axes
         */
   EIB7_AXIS axis[NUM_OF_AXIS];      /* axes array
         */
   char fw_version[20];              /* firmware version string
         */
   int enc_axis;                     /* actual axis index
         */
   int enc_type;                     /* encoder type
         */
   int i;


#ifdef Linux
   signal(SIGINT, CtrlHandler);
```

```
  signal(SIGTERM, CtrlHandler);
#endif


 char hostname[12] = {'1','9','2','.','1','6','8','.','1','.','2','\0'}; // this is the lan IP address
 enc_axis=0;  // this selects the first encoder axis
 enc_type=1;  // This selects 1vpp as the encoder output

  /* open connection to EIB */
  CheckError(EIB7GetHostIP(hostname, &ip));
  CheckError(EIB7Open(ip, &eib, EIB_TCP_TIMEOUT, fw_version,
      sizeof(fw_version)));


  /* get axes array */
  CheckError(EIB7GetAxis(eib, axis, NUM_OF_AXIS, &num));


  /* initialize selected axis */
  /* 1 Vpp */

     CheckError(EIB7InitAxis(axis[enc_axis],
                 EIB7_IT_Incremental,
                 EIB7_EC_Linear,
                 EIB7_RM_None,       /* reference marks not
                     used */
```

```
                    0,                    /* reference marks not
                        used */
                    0,                    /* reference marks not
                        used */
                    EIB7_HS_None,
                    EIB7_LS_None,
                    EIB7_CS_CompActive,   /* signal compensation
                        on    */
                    EIB7_BW_High,         /* signal bandwidth:
                        high    */
                    EIB7_CLK_Default,     /* not used for
                        incremental interface */
                    EIB7_RT_Long,         /* not used for
                        incremental interface */
                    EIB7_CT_Long          /* not used for
                        incremental interface */
                    ));
    while (true)
    {
    /* call polling loop function */
    PollPosition(axis[enc_axis],enc_type, encpos);
    pos = encpos;
    //cout << pos << endl;

    }

    /* close connection to EIB */
```

```
    EIB7Close(eib);

    m5.lock();

    pos = encpos;

    m5.unlock();

}



int picomotor(int handle, int n,int t, int Dist,double Spd,
    double dir){

int err;

            // Open first found LabJack
            //handle = OpenOrDie(LJM_dtANY, LJM_ctANY, "LJM_idANY
                ");
            char * name2;

      // identifies the active IO port FI01 as the port being
          commanded
      name2 = "FIO1";

      // Set DIO state on the LabJack
```

```
        err = LJM_eWriteName(handle, name2, dir);
        ErrorCheck(err, "LJM_eWriteName");

        for (n; n< t; n = n+1){

        // Set up for setting DIO state
        double value = 1; // Output state (0 = low, 1 = high)
        char * name;

    name = "FIO0";

        // Set DIO state on the LabJack
        err = LJM_eWriteName(handle, name, value);
        ErrorCheck(err, "LJM_eWriteName");

    struct timespec tim, tim2;
    tim.tv_sec = 0;
    tim.tv_nsec = Spd;

    if(nanosleep(&tim , &tim2) < 0 )
    {
       printf("Nano sleep system call failed \n");

    }

    value = 0;

```

```
        err = LJM_eWriteName(handle, name, value);
        ErrorCheck(err, "LJM_eWriteName");
        }



int picomotor2(int handle, int n,int t, int Dist,double Spdy,
    double diry){
//int handle;
int err;


        // Open first found LabJack
        char * name3;

    // identifies the active IO port FI01 as the port being
        commanded
    name3 = "FIO3";

    // Set DIO state on the LabJack
        err = LJM_eWriteName(handle, name3, diry);
        ErrorCheck(err, "LJM_eWriteName");

        for (n; n< t; n = n+1){

        // Set up for setting DIO state
        double value = 1; // Output state (0 = low, 1 = high)
```

```
        char * name4;

    name4 = "FIO2";

        // Set DIO state on the LabJack
        err = LJM_eWriteName(handle, name4, value);
        ErrorCheck(err, "LJM_eWriteName");

   struct timespec tim, tim2;
   tim.tv_sec = 0;
   tim.tv_nsec = Spdy;

   if(nanosleep(&tim , &tim2) < 0 )
   {
      printf("Nano sleep system call failed \n");

   }

   value = 0;

        err = LJM_eWriteName(handle, name4, value);
        ErrorCheck(err, "LJM_eWriteName");
        }
        }

int temp( int handle)
{
```

```
  while (true)
      {

      int err;
      //int handle;

      // Set up for reading AIN value
      double temp = 0;

      const char * NAME = "AIN0";


        struct timespec tim, tim2;
tim.tv_sec = 0;
tim.tv_nsec = 100000000;

if(nanosleep(&tim , &tim2) < 0 )
{
    printf("Nano sleep system call failed \n");

}

      // Read AIN from the LabJack
      err = LJM_eReadName(handle, NAME, &temp);
      ErrorCheck(err, "LJM_eReadName");
```

```
temperature=((13.582 -sqrt((184.470724)
    +.01732*(2230.8-temp*1000)))/-.00866)+30;
}
}


void vertcontrol(int handle)
{
int n=0;
int t;
int Dist;
double Spdy=99999999;
double diry;
sleep(10);
double locyzero=(locy+locy2)/2;

cout <<"the y zero is" << locyzero << endl;

while (true)
{ double locydelta=((locy+locy2)/2)-locyzero;
if (locy == 0)
{quitnum=1;
break;}

else
```

```
        if (locy2 == 0)
        {quitnum=1;
        break;}

        else

        if (locydelta < -3)
        {diry = 0;}
        else
        if (locydelta > 3)
        {diry = 1
;}

        if (locydelta > 3)
        {t=Dist=locydelta*1;
        picomotor2(handle,n,t,Dist,Spdy,diry);
    }

        else

    if (locydelta < -3)
        {t=Dist=-locydelta*1;
        picomotor2(handle,n,t,Dist,Spdy,diry);
        }

        else
```

```
        {cout << "on target" << endl;}

        cout << locydelta << endl;
        cout << diry << endl;

        //picomotor2(handle,n,t,Dist,Spdy,diry);
        sleep(3);


  if (quitnum == 1)
  {break;}

        }
        }
void encavg()
{

while (true){
double tot2;
double rep2=1;
long int curpos;
  tot2=0;
    for (double k = 1; k <= rep2; ++k){


    curpos=encpos-zero;
    tot2=tot2+curpos;
```

```

    }
    avg2 = tot2/rep2;

}
}

void data()
{

//mutex m3;
//mutex m4;
double avg;
double tot;
double rep =100;     //this is normally at 2000
//long int pos;

double avg3;
double tot3;

double convfact;
double combavg;

ofstream campos;

campos.open("camposdata.csv");
campos << "time" << "," << /*"averaged camera measurement 1"
```

```
<< "," << "averaged camera measurment 2"<< ","<< "non
averaged camera 1" << "," << */ "encoder avg" << "," << "
measurment 1 and 2 average" << "," << "measurement 1 y" <<
"," << "measurement 2 y" << "," << "temperature" << ","<<
"averaged fit x"<< endl;


// averaging the first cross correlation position
____________________________________________
while (true)
{
double comb_loc=loc+fitx1;

auto timenow =
    chrono::system_clock::to_time_t(chrono::system_clock::
        now());

    tot=0;
    for (double i = 1; i <= rep; ++i){

  struct timespec tim, tim2;
  tim.tv_sec = 0;
  tim.tv_nsec = 10000000;

  if(nanosleep(&tim , &tim2) < 0 )
  {
    printf("Nano sleep system call failed \n");
```

```
   }
    tot=tot+comb_loc;

    }
    avg = tot/rep;

// averaging the second cross correlation position

        tot3=0;
        double comb_loc2=loc2+fitx2;

        for (double i = 1; i <= rep; ++i){

        struct timespec tim, tim2;
        tim.tv_sec = 0;
        tim.tv_nsec = 10000000;

        if(nanosleep(&tim , &tim2) < 0 )
      {
        printf("Nano sleep system call failed \n");
      }

      tot3=tot3+comb_loc2;

    }
    avg3 = tot3/rep;
```

```
//

    m4.lock();

     convfact=14.325;

     combavg=((avg3+avg)/2)*convfact;

  campos << ctime(&timenow) << "," /*<< avg3 << "," << avg <<
       "," << loc << ","*/ << avg2 << "," << combavg << "," <<
       locy << "," << locy2 << "," << temperature << "," <<
       avgfitx<< endl;

  m4.unlock();
   sleep(1);


cout<< avg <<" , "<< avg3<< endl;
cout << "combined average:  " << combavg << endl;
   if (quitnum == 1)
   {break;}


  }
```

```
}
```

APPENDIX C: Encoder Uncertainty Analysis

$r_i$ = initial length of flexure lever arm

$\Delta r$ = change in length of flexure lever arm

$h_i$ = initial height of flexure angle adjustment screw

$r_{wire}$ = initial length of scale

$\Delta h$ = the change in height of flexure angle adjustment screw

$T$ = environmental temperature withing environmental enclosure

$\alpha_{steel}$ = coefficient of thermal expansion of steel

$\theta_1$ = initial angle of flexure hinge

$\theta_2$ = angle of flexure hinge after thermal expansion

$\Delta_\theta$ = change in angle of flexure hinge

$\sigma r_i$ = uncertainty of initial length of flexure lever arm

$\sigma h_i$ = uncertainty of initial height of flexure angle adjustment screw

$\sigma\alpha_{steel}$ = uncertainty of the coefficient of thermal expansion of steel

$\sigma r_x$ = uncertainty of calculated x position

$\sigma\theta$ = uncertainty of flexure hinge angle

$\sigma x$ = uncertainty of x position measured by camera

$\sigma r_{scale}$ = uncertainty in the initial length of scale

$$\Delta r = \alpha_{abs} r_i \Delta T \tag{C.1}$$

$$\Delta h = \alpha_{abs} h_i \Delta T \tag{C.2}$$

The change of angle $\Delta_\theta$ due to change in temperature can be calculated using equations (C.3) through (C.5).

$$\theta_1 = tan^{-1}\left(\frac{h_i}{r_i}\right) \tag{C.3}$$

$$\theta_2 = tan^{-1}\left(\frac{h_i + \Delta h}{r_i + \Delta r}\right) \tag{C.4}$$

$$\Delta_\theta = \theta_2 - \theta_1 \tag{C.5}$$

The uncertainty of $\Delta r$, $\Delta h$, and $\Delta wire$ can be calculated using equations (C.6), through (C.8)

$$\sigma_{\Delta r} = \sqrt{(\alpha_{steel}\Delta T \sigma r_i)^2 + (r_i \Delta T \sigma_{\alpha steel})^2 + (r_i \alpha_{steel} \sigma_{\Delta T})^2} \tag{C.6}$$

$$\sigma_{\Delta h} = \sqrt{(\alpha_{steel}\Delta T \sigma h_i)^2 + (h_i \Delta T \sigma_{\alpha steel})^2 + (h_i \alpha_{steel} \sigma_{\Delta T})^2} \tag{C.7}$$

$$\sigma_{\Delta r_{scale}} = \sqrt{(\alpha_{steel}\Delta T \sigma r_{scale})^2 + (r_{scale} \Delta T \sigma_{\alpha steel})^2 + (r_{scale} \alpha_{steel} \sigma_{\Delta T})^2} \tag{C.8}$$

Equations (C.9) through (c.11) are used to calculated the uncertainty of the angle of between horizontal axis of the camera and the translation axis of the camera, which is shown in equation (8.16).

$$\frac{\partial \theta}{\partial h_i} = \frac{r_i + \Delta r}{\Delta h^2 + 2\Delta h h_i + h_i^2 + (r_i + \Delta r)^2} + \frac{h_i}{r_i^2 + h_i^2} \tag{C.9}$$

$$\frac{\partial \theta}{\partial \Delta h} = \frac{r_i + \Delta r}{\Delta h^2 + 2\Delta h h_i + h_i^2 + (r_i + \Delta r)^2} \tag{C.10}$$

$$\frac{\partial \theta}{\partial r_i} = \frac{h_i + \Delta h}{(r_i + \Delta r)^2 \left(\frac{(h_i + \Delta h)^2}{(r_i + \Delta r)^2} + 1\right)} + \frac{h_i}{h_i^2 + r_i^2} \tag{C.11}$$

$$\frac{\partial \theta}{\partial \Delta r} = \frac{h_i + \Delta h}{(r_i + \Delta r)^2 \left(\frac{(h_i + \Delta h)^2}{(r_i + \Delta r)^2} + 1\right)} \tag{C.12}$$

$$\sigma_\theta = \sqrt{\left(\frac{\partial\theta}{\partial h_i}\sigma h_i\right)^2 + \left(\frac{\partial\theta}{\partial \Delta h}\sigma \Delta h\right)^2 + \left(\frac{\partial\theta}{\partial r_i}\sigma r_i\right)^2 + \left(\frac{\partial\theta}{\partial \Delta r}\sigma \Delta r\right)^2} \tag{C.13}$$

The equation for the uncertainty of the $x$ position, as measured by the camera, is shown in equation (C.14)

$$\sigma_x = \sqrt{(cos(\Delta\theta)\sigma_{rx})^2 + (cos(\Delta\theta)\sigma\Delta\theta)^2 + ((-r_x sin(\Delta\theta) - \Delta r_x sin(\Delta\theta))\sigma\theta)^2} \tag{C.14}$$