Automated Vernier Scale Readout via Image Processing for Detector Positioning in Hall C at Jefferson Laboratory

Ralph Marinaro
Faculty Advisor: Dr. Edward Brash

April 2019

1 Abstract

Image processing is a method of information extraction which, for the purposes of this project, is used for calculating the angular position of a detector in Hall C at Jefferson Laboratory. The angular position of the detector relative to the electron beamline is an important value for data acquisition regarding the position and trajectory of resultant scattered particles during experimentation. The angular position of the detector can be calculated through a series of steps involving the application of physical algorithms used to manipulate detector angle images which are extracted from CODA data files. The angular positions calculated by an image processing program, within a certain error, can be compared to angle values output by an encoder device and the angle values that are visible in the images.

2 Introduction

The goal of this project is to create an automatic, precise tool for measuring the angle at which a detector in Hall C at Jefferson Laboratory (JLab) is set, measured in relation to the beam line. The experimental hall houses two sets of detectors, the Super High Momentum Spectrometer (SHMS) and the High Momentum Spectrometer (HMS), which rotate about a point central to a circular track upon which the wheels of the detector sets are fixed to move. At the center of this circular track is the target. The electron beam used at JLab directs electrons moving at relativistic speeds towards the target and the resultant collisions scatter particles in the direction of the detectors. The angles at which these detectors are set during experiment runs should be set for maximal qualitative observing and recording of resultant particle scattering. Knowing these angle measurements down to the hundredth of a degree also provides accurate values used for calculating different properties of the scattered particles such as energy and position. A diagram of the same layout for an experimental hall can be seen in Figure 1.

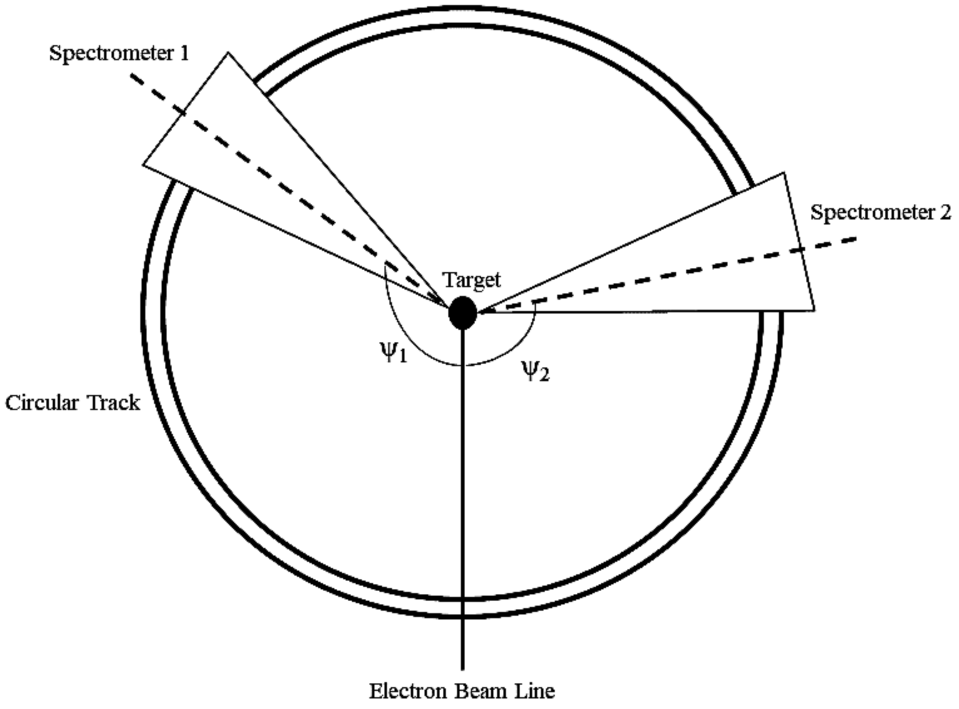

Figure 1: Basic layout of an experimental hall.

In Figure 1, ψ_1 and ψ_2 are the angles for the HMS and SHMS respectively with each angle measured to the detector center. Attached to each detector center is a camera which takes a picture of the angle measure marked on the floor. Each picture contains the larger decimal place angle measure written out and a Vernier scale to measure the hundredth decimal place. An example photo can be seen in Figure 2.

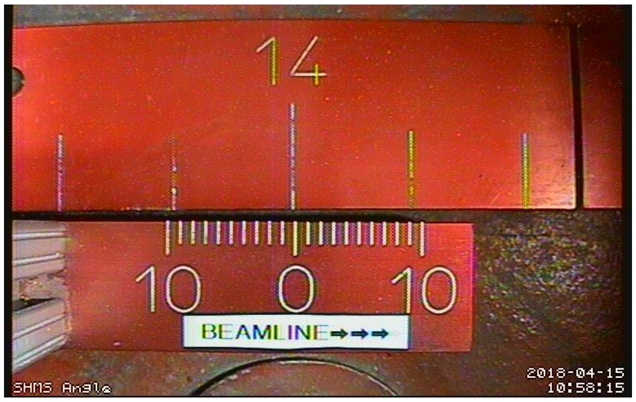

Figure 2: Example angle photo with Vernier scale

The large angle value is not necessarily always centered within the range of the Vernier scale as seen in Figure 3. This will not be a problem since the numbers in the image are not extracted. Instead the large angle markings and the Vernier scale markings provide the necessary data for calculating the angle measure down to the hundredth of a degree.

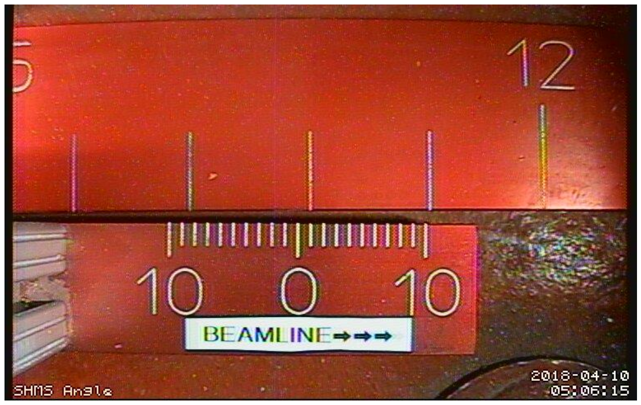

Figure 3: Example angle photo with large angle value offset from center.

The current process for recording the detector angle measurement uses a device called an encoder. The encoder moves on the circular track with the detector and outputs the angle measurement based on the change in rotational position detected. As the encoder moves along the track slight slips between the encoder and the track over time decrease accuracy. The disconnects between the track and the encoder are small enough that the angle measure maintains accuracy down to the tenth of a degree, but no further. This project purpose is to create an image processing program (IPP), using the Python image processing libraries, which will be designed for use at JLab. The IPP can be run continuously during experiments as new CODA data files are generated allowing for automated recording of the angle measure down to the hundredth of a degree with an accuracy that does not decrease over time. Using the angle measure down to the tenth of a degree from the encoder, the IPP extracts the hundredth decimal value from the angle photos by focusing on the Vernier scale and adds that value to the angle value measured to the tenth of a degree provided by the encoder.

3 Theory

The physical theory applied during image processing is a Fourier transformation algorithm which transforms from spatial domain coordinates to frequency domain coordinate as shown in Equation 1.

$$F(k,l) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j)e^{-2\pi(\frac{ki}{N}+\frac{lj}{N})} \qquad \text{(Eq. 1)}$$

The function $F(k,l)$ is the Fourier transform of the spatial function $f(i,j)$ where *k* and *l* are the wavelength and frequency coordinates and *i* and *j* are the x and y coordinates for each pixel in the image This equation is for an image of *N* by *N* pixels, but any rectangular image can be analyzed using the same equation by simply adjusting the *N* values. Equation 2 shows the inverse Fourier transform going from the frequency domain back to the spatial domain.

$$f(a,b) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} F(k,l)e^{-2\pi(\frac{ka}{N}+\frac{lb}{N})} \qquad \text{(Eq. 2)}$$

The variables *a* and *b* are the x and y coordinates in the spatial domain after the inverse transformation. The Fourier transform changes the pixel mapping of the image from using position coordinates to coordinates of wavelength and frequency.

The hundredth angle measure output by the IPP is calculated using a linear fit which is applied to the x locations, measured in pixels, of each Vernier scale marking. The intercept with the y-axis provides the number of pixels in x at which the large angle marking is offset and from this value the hundredth angle value is calculated. The fit to this set of values also allows for the IPP to generate a covariance matrix between the slope and y intercept values as shown in Equation 3.

$$\sigma^2(m,b) = \begin{bmatrix} \sigma^2(m) & \sigma(m,b) \\ \sigma(m,b) & \sigma^2(b) \end{bmatrix} \qquad \text{(Eq. 3)}$$

$$\sigma(b) = \sqrt{\sigma^2(b)} \qquad \text{(Eq. 4)}$$

In Equation 3, $\sigma^2(m,b)$ is the covariance matrix for the slope and y intercept. The error for the intercept value is calculated by taking the square root of $\sigma^2(b)$ as shown in Equation 4. The error for the y intercept is the error given for the hundredth angle value output by the code and is also equal to the error for the total angle value calculated using the value provided by the encoder,

4 Methods

The IPP goes through a series of steps to extract, manipulate, and analyze an image. The IPP first asks for the run number of the image to be analyzed and uses the given path to find the corresponding CODA data file for the given run number. The code then analyzes the first million lines of the data file to find the encoder angle value and extract the angle image. Once both the image and encoder angle value are extracted, the code goes through a series of image manipulation commands which are described in the Appendix. These manipulation commands help reduce noise and improve the quality of the photo as seen in Figure 4 and Figure 5.

In Figure 4, the graph of the pixels in the x-y plane is plotted against the scaled frequency value given for the individual pixels. The grey-scaled image has a large amount of visible noise which is noticeably reduced in the second image of Figure 4. This reduction is then combined with an improvement to the contour edges in the first image of Figure 5 and this improved image is then binarized for the second image of Figure 5. Figure 7 provides a better view from above of the final mapping overlaid on the grey-scaled image which shows well defined contours for both the large angle and Vernier scale markings.

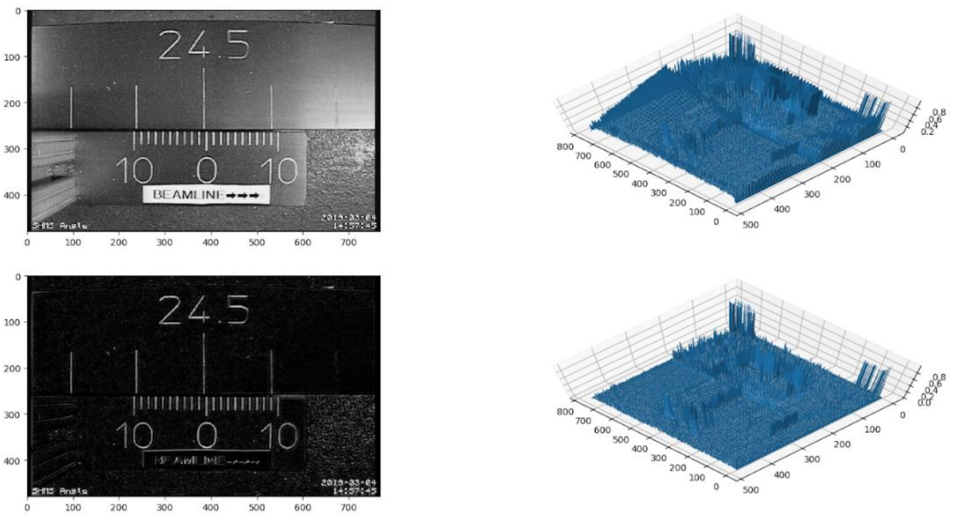

Figure 4: Grey-scaled image and noise reduction.

The image is then binarized such that the markings for both the Vernier scale and the larger angle values are white and the rest of the photo is black. Some other sections of the image may be white other than the markings, but the code is given pixel value parameters in x and y coordinates defining regions in which the IPP looks for contours. These image sections are called regions of interest and the contours are the white sections representing the angle markings as seen in Figure 5.

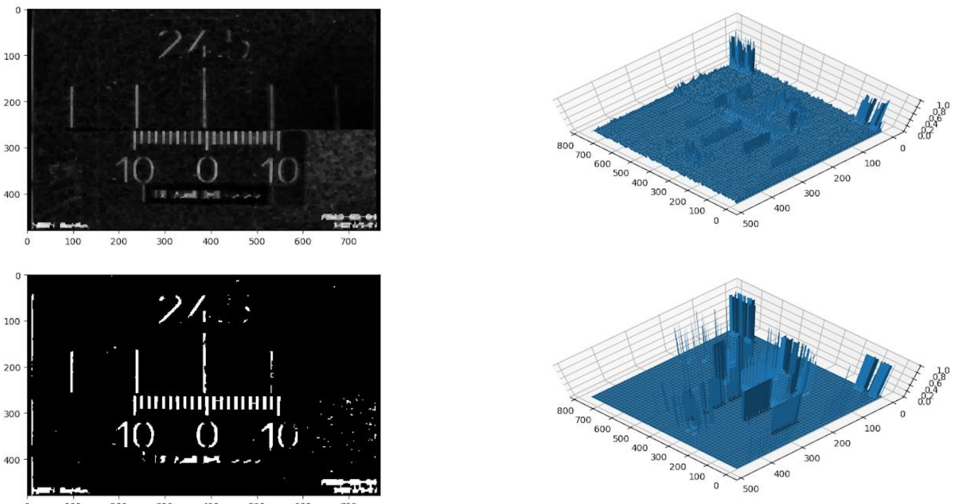

Figure 5: Erosion, dilation, and binarization of angle image.

The contour locations are calculated using a Fourier transform which determines where large changes in frequency occur between the white and black pixels. The frequency changes are then transformed back to the spatial domain so the that x and y coordinates for all contours within the regions of interest are stored in an array. These contours can be seen overlaid on the original image in Figure 6.

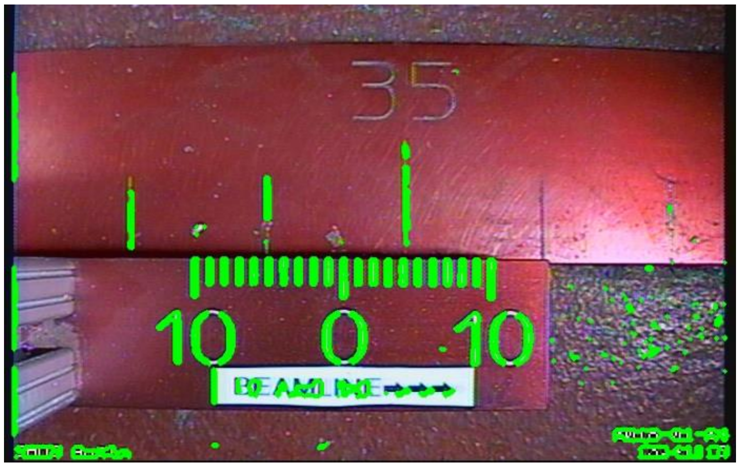

Figure 6: Original image with contours overlaid.

The contours for the larger angle markings are noticeably misshapen in comparison to the Vernier scale markings. To get the most well-defined value for the pixel coordinate, the largest continuous contour available for the larger angle marking is used. The contours can also be seen overlaid on the greyscale image to provide a visual represent of the final mapping for the Vernier scale and large angle markings as seen in Figure 7.

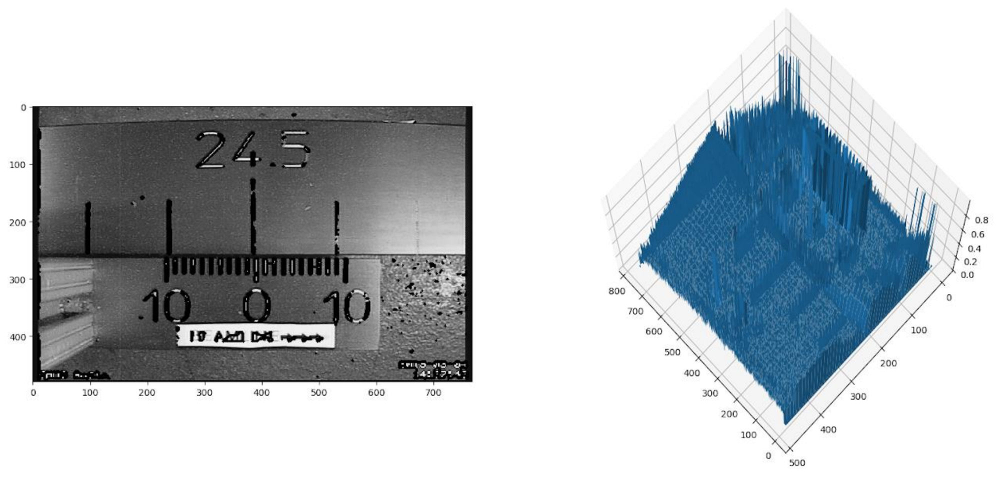

Figure 7: Grey-scaled image with contour overlaid.

The x coordinate positions for each of the twenty-one Vernier scale marking are used to calculate the hundredth angle value by applying a linear fit to the x values and the covariance matrix from this fit also provides the error. This linear fit can be seen in Figure 8.

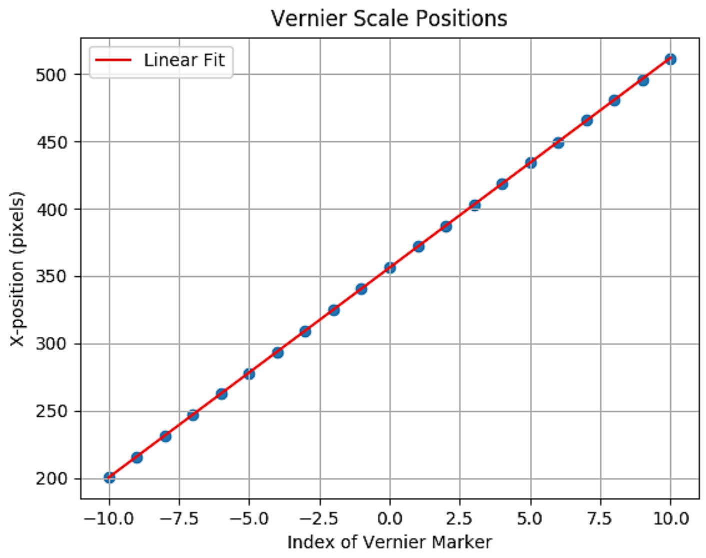

Figure 8: Graph of linear fit for Vernier Scale contours

The hundredth angle value calculated by the IPP is then added to the large angle value given by the code and a final angle value is output by the code. These outputs are the IPP data set for the angle values that are compared to the angle values given by the encoder and the angle values visible in the images. In a few instances, an observation was made that contours can overlap as seen in Figure 9 between the 0.01-0.02 values and the 0.06-0.07 values.

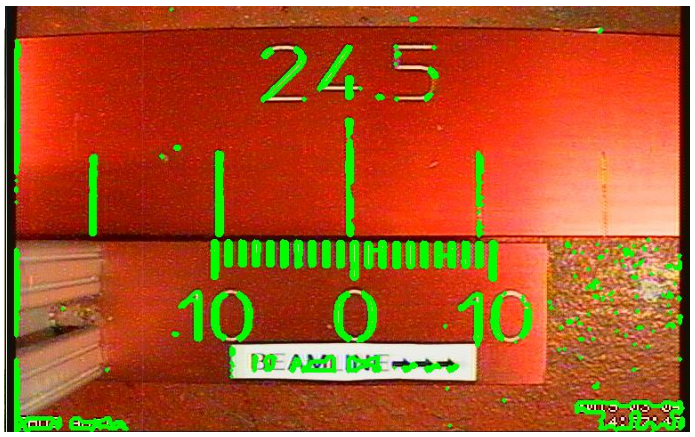

Figure 9: Original Image with contour overlap.

A section of the IPP takes this overlap into consideration and uses adjacent contour x coordinate values to approximate the value for the x coordinate for the overlapped contour. The resultant graph for the x coordinates value in this case is shown in Figure 10.

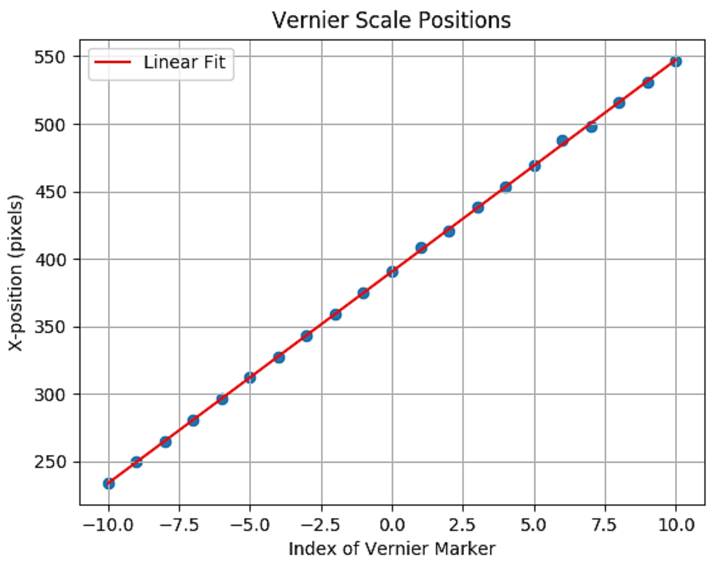

Figure 10: Graph of linear fit for contour overlap

A comparison between Figure 8 and Figure 10 shows that the two overlapped contours in Figure 10 are slight offset from the linear fit while the contours in Figure 8 are a perfect linear fit without contour overlap. This offset in Figure 10 ultimately does not greatly affect the linear fit since the x coordinate contour values are evenly offset from the linear fit such that the two offsets cancel each other out.

5 Data

The data for this project is comprised of over 140 data points representing the values output by the IPP for a series of CODA data files. These CODA data files correspond to experimental runs taken at Jefferson Lab for several different detector angle settings. This data set is compared to two other data sets which are the angle values provided by the encoder and the angle values that are visibly measurable by hand from the images. The following graphs in Figures 9-12 show the angle values given by the IPP, encoder, and image for four different angle values for which there are consecutive CODA data files. The encoder data points are plotted in red, with the image and IPP data points plotted in gold and blue respectively.

Legend for plots in Figures 9-12.

▲	-	Encoder
◆	-	IPP
■	-	Image

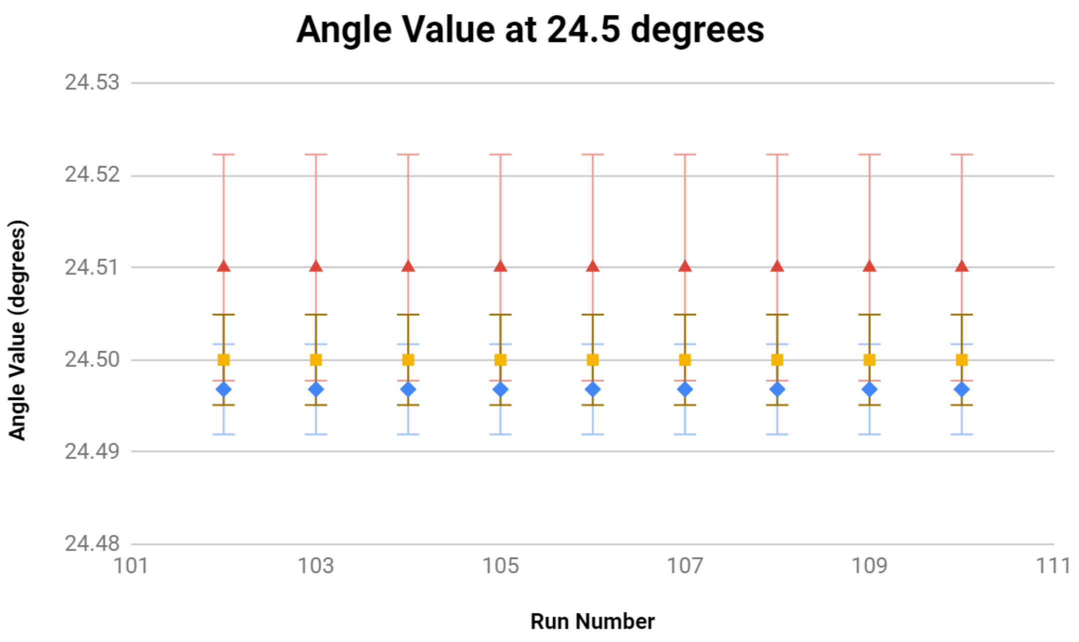

Figure 9: Plot of angle values at 24.5 degrees.

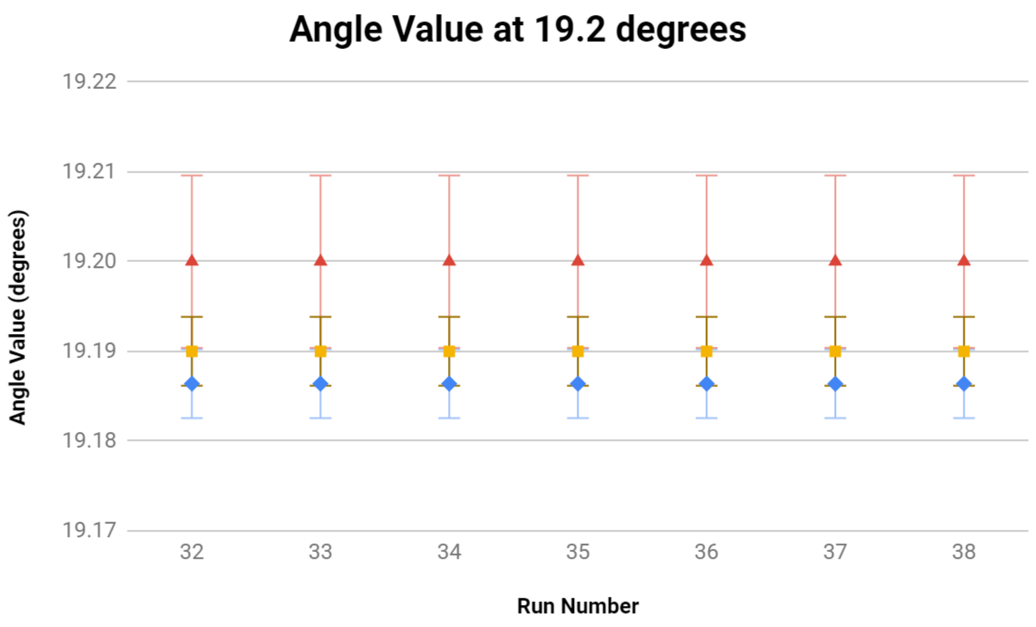

Figure 10: Plot of angle values at 19.2 degrees.

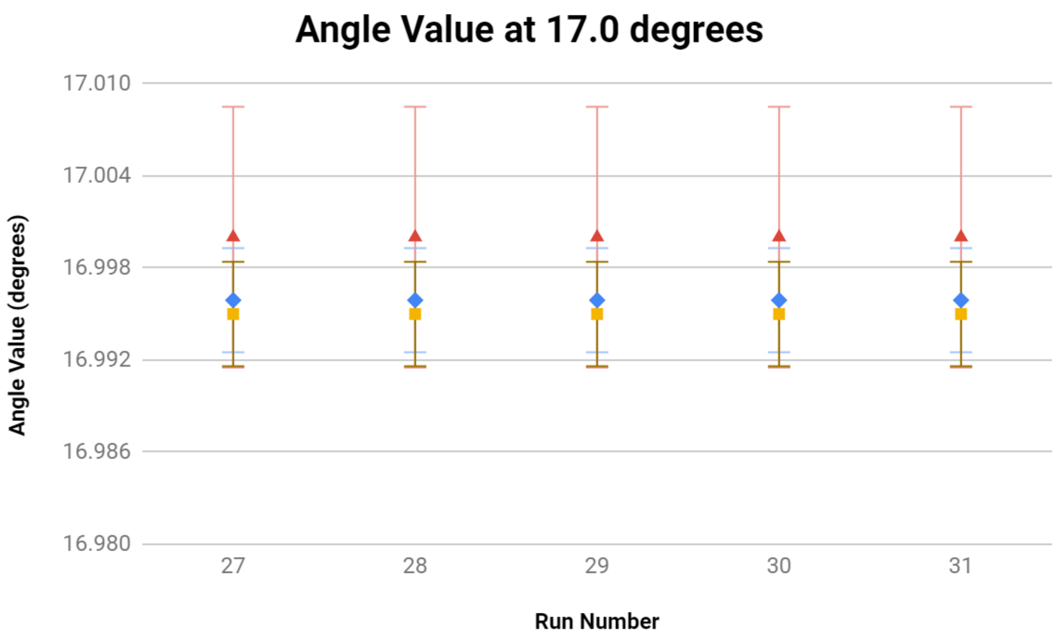

Figure 11: Plot of angle values at 17.0 degrees.

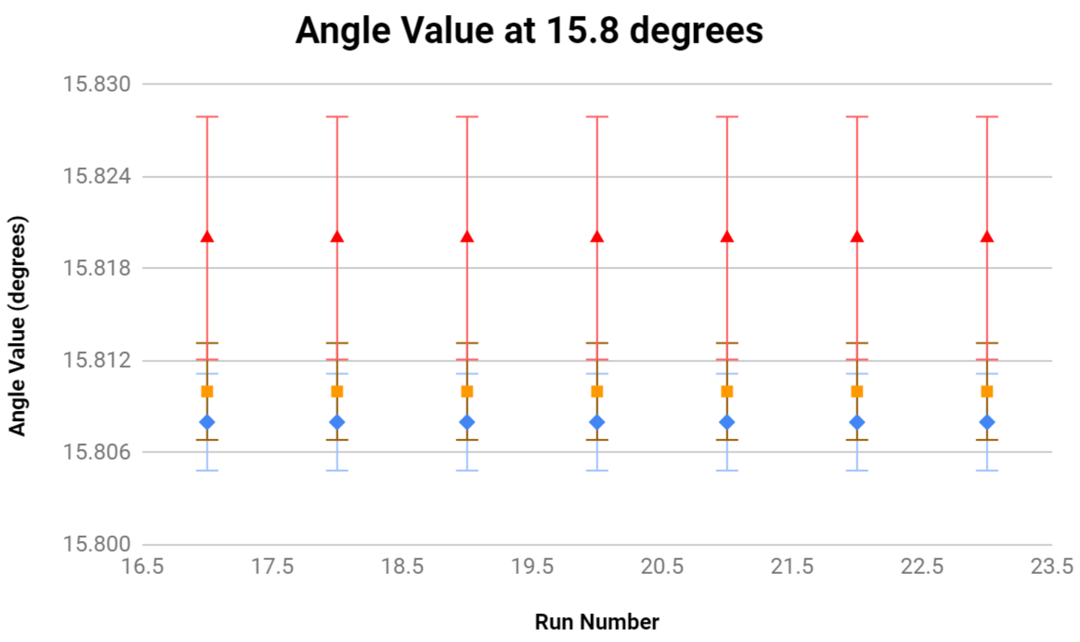

Figure 12: Plot of angle values at 15.8 degrees.

Each of these four graphs in Figures 9-12 show that for consecutive data runs with the same angle value, the IPP outputs consistent angle values down to the hundredth of a degree. The angle values visible in the images are also consistently within the error of the IPP angle value which shows that the IPP is an accurate tool for measuring the angle value within an error of ±0.01 degrees. The encoder angle values are consistently larger in value than the image and IPP values in Figures 9-12, but that is not necessarily true throughout the data. There is only one data point which currently has the encoder value equal to the IPP value and this is because the IPP failed to recognize contours for the large angle markings. The default output in this case is the encoder angle value.

6 Conclusions

An ideal IPP would be able to output angle values which are consistent for individual angle values over a series of runs and these outputs should be within an error of the angle measure visible in the image. The ideal IPP would also improve upon the current system by automatically generating angle values while experiments continue to run, and the accuracy of these angle values would not decrease over time. With all these performance expectations taken under consideration, the IPP created for this project meets all the original specifications. As seen in the data provided, the IPP improves upon the angle values output by the encoder using image processing. This improvement can be seen in the average difference between encoder and image values versus IPP and image values. The average difference between the encoder and image values is 0.012 degrees while the average difference between IPP and image values is 0.003 degrees. This shows that the IPP increases the accuracy of the angular position on average by a factor of four. A similar conclusion can be made form looking at the average percent difference between the encoder and image values which is 5.7% versus the average percent difference between the IPP and image values which is 1.5%. This again shows the increased accuracy of the IPP when measuring the angle value.

One discrepancy in the data shows the angle value output by the IPP to be exactly equal to the encoder angle value. This is because the IPP was not able to identify any contours for the large angle markings in the image. Upon further observation of the image, the lighting in the image appears to not be as beneficial for identifying differences between angle markings and the rest of the image. This is most likely due to some change in the lighting surrounding the place in the circular track such as an object casting a shadow. The IPP does take this sort of case into account and defaults to output the encoder angle value.

For the purposes of this project, images for the SHMS were only used. Further research can be completed for the HMS and possibly for the detectors in Hall A at Jefferson Lab. Attempts were made at analyzing photos for the HMS, but the images for the HMS are noticeably different from the SHMS. The HMS images are taken much further away from the angle markings and the IPP is not able to define any contours for the significantly smaller Vernier and large angle values. If the IPP were to be used for the HMS, the best course of action would be to first adjust the camera closer to the angle markings and possibly improve the lighting to be a little less bright since the images for the HMS currently appear washed out by a large amount of brightness. Past attempts have been made at improving the angle measurement process at JLab, but none currently utilize image processing for a comparison to this project.

7 Appendix

Opening is just another name of erosion followed by dilation using the morphology function used in the IPP for this project.

```
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
```

Closing is the reverse of Opening, dilation followed by erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object.

```
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

Morphological Gradient is the difference between dilation and erosion of an image. The result will look like the outline of the object.

```
gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)
```

Top Hat is the difference between the input image and the Opening of the image.

```
tophat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)
```

Structure elements are manually created with help from Numpy with most elements being rectangular shapes. But in some cases, an elliptical or circular shaped kernel may be needed. For this purpose, OpenCV has a function, cv2.getStructuringElement() which is passed the shape and size of the kernel providing the desired kernel.

```
sqkernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5,5))
```

The above command gives a five by five kernel which can be used for any of the pervious image manipulation commands. OpenCV provides three types of gradient filters or High-pass filters, Sobel, Scharr and Laplacian. The Sobel operator is a joint Gausssian smoothing plus differentiation operation which is more resistant to noise. The direction of derivatives to be taken must be specified as either vertical or horizontal. The the size of kernel can also be specified by the argument ksize. If ksize = -1, a 3x3 Sobel filter is used.

```
sobel = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)
```

Thresholding works such that if a pixel value is greater than a threshold value, it is assigned one value, else it is assigned another value. The function used is cv2.threshold. The first argument is the source image, which should be a grayscale image. The second argument is the threshold value which is used to classify the pixel values. The third argument is the maxVal which represents the value to be given if pixel value is more than the threshold value. OpenCV provides different styles of thresholding and it is decided by the fourth parameter of the function which could be one of five different functions.

Two outputs are obtained with the first one being the retval and the second output is the thresholded image.

```
thresh = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
```

The provided image manipulation commands listed in this appendix give a comprehensive review of all the image manipulation commands found in the IPP.

Bibliography

[1] Rosebrock, A. (2017). Credit Card OCR with OpenCV and Python. PyImageSearch. Retrieved September 4, 2018 from https://www.pyimagesearch.com/2017/07/17/credit-card-ocr-with-opencv-and-python/

[2] The Python Software Foundation. (2018). Python/C API Reference Manual. Python Software Foundation. Retrieved September 10, 2018 from https://docs.python.org/2/c-api/