

Development of a Technique for Sub-nm Measurements of 1-D Motion and its  
Application to Mems

By

Ryan Adderson

Submitted in partial fulfilment of the requirements  
for the degree of Master of Applied Science

Dalhousie University  
Halifax, Nova Scotia  
March 2019

© Copyright by Ryan Adderson, 2019

## Table of Contents

List of Tables .....	v
List of Figures .....	vi
ABSTRACT.....	xi
List of Abbreviations of Symbols Used .....	xii
1. Introduction .....	1
1.1. Objective .....	1
1.2. Scope and Limitations .....	2
2. Literature Review.....	3
2.1. MEMS Devices.....	3
2.2. Optical Displacement Measurements.....	3
2.2.1. Why are Optical Measurements Used?.....	3
2.2.2. Digital Image Correlation .....	4
2.2.3. MEMS Optical Measurements .....	6
3. Existing MEMS Devices .....	10
3.1. PolyMUMPs.....	10
3.2. Thermal Actuators.....	12
4. Existing Yamahata Algorithm.....	16
4.1. Image Processing.....	16
4.2. Reference Structures.....	20
4.3. Typical Chevron Motion .....	22
4.4. Accuracy, Precision, and Resolution .....	26
5. Cross Correlation.....	28
5.1. Theory .....	28

5.2.	SSSIG & MIG .....	30
5.3.	DIC errors .....	32
6.	Cross Correlation Simulation .....	34
6.1.	Measuring Sub-Pixel.....	34
6.2.	Noise.....	41
6.3.	Offset.....	45
6.4.	Slope.....	47
6.5.	Uneven Illumination.....	49
6.6.	Summary .....	51
7.	Experimental Setup.....	52
7.1.	Microscope.....	52
7.2.	Camera .....	54
7.3.	MEMS Chip.....	55
7.4.	Cross-Correlation Algorithm: XCorr .....	58
8.	Optical Tests.....	63
8.1.	Yamahata vs. Xcorr (Chevron).....	63
8.1.1.	Test Procedure .....	63
8.1.2.	0 Voltage (Off) Test.....	64
8.1.3.	1 Volt (On) Test .....	68
8.1.4.	1.0 to 1.1 Volt STEP Test .....	71
8.1.5.	1.00 to 1.01 Volt STEP Test .....	73
8.2.	Measurement Noise.....	77
8.3.	Impact of Colour.....	79
8.4.	Impact of Number of Periods, Bar vs. Dot, Size in Yamahata / XCorr.....	82

8.4.1.	Why is this Important?.....	82
8.4.2.	Test Setup .....	82
8.4.3.	Bar Number.....	83
8.4.4.	ROI Height .....	88
8.4.5.	Dimples .....	92
8.4.6.	Non-Periodic Structures.....	95
9.	Conclusions & Future Work .....	98
	Appendix A-1: XCorr MATLAB Program .....	100
	Appendix A-2: Image Processing MATLAB Function.....	107
	Appendix A-3: Column Average MATLAB Function .....	109
	Appendix A-4: Cross Correlation MATLAB Function .....	110
	Appendix B-1: Image Capture Only LabVIEW VI .....	112
	Appendix B-2: Function Generator and Image Capture LabVIEW VI.....	113
	References .....	116

## List of Tables

Table 1: Comparison of several displacement measurement methods.....	9
Table 2: Simulated displacement measurements at varying resolutions. ....	40
Table 3: Error in measurement between expected and measured displacement at various resolutions.....	40
Table 4: Expected deviation of measurements of simulated data with noise. ....	43
Table 5: Measured displacements of simulated data with vertical offset. ....	47
Table 6: Measured displacements of simulated data with a slope. ....	49
Table 7: Measured displacements of simulated data with an uneven slope. ....	50
Table 8: The average measurement and standard deviation at each step in the 1.00 – 1.01 mV test as measured by the XCorr. ....	76
Table 9: Results of measurements taken using different colour scales. ....	81
Table 10: Measured standard deviation for XCorr and Yamahata algorithms as number of periods is varied.....	86
Table 11: Measured standard deviations for the XCorr and Yamahata as the height of the ROIs were varied. ....	90
Table 12: Comparison of XCorr and Yamahata results for bars and dimples.....	93
Table 13: Measurement standard deviations for the XCorr and Yamahata for different ROIs, and the ratio. ....	96

## List of Figures

Figure 1: A representation of a speckle pattern which would be tracked by a traditional DIC algorithm. ....	5
Figure 2: A representation of a speckle pattern which would be tracked on a MEMS device. ....	6
Figure 3: PolyMUMPS layers.....	11
Figure 4: A simple cantilever beam composed of Poly 0 (orange) and Poly 1 (red) layers in a PolyMUMPS. MEMS device, as seen from the top (above) and side (below). ....	12
Figure 5: L-Edit model of a chevron actuator (above) and photo of an actuator on MEMS chip (below). The direction of motion is indicated.....	13
Figure 6: A double thermal actuator design in L-Edit (left) and on the MEMS chip (right). ....	14
Figure 7: RGB image reformatted to greyscale. ....	17
Figure 8: A greyscaled image of a MEMS chip (left), and a region of interest used in measurements (right). ....	18
Figure 9: The column averaged values, showing 6 zero crossings (green) and 5 periods (blue) which will be used in calculations. ....	19
Figure 10: Normalized FFT of a 1D profile indicating a 10 $\mu\text{m}$ spatial wavelength.....	20
Figure 11: Component on a MEMS chip, with moving ROI (orange), and reference ROI (purple). ....	21
Figure 12: Measurements of moving and reference structures, as well as their difference.....	22
Figure 13: A MEMS device using chevron actuators (left) and a zoom in of the device (right). ....	23
Figure 14: The ROI used to determine the motion created by the actuator. ....	24
Figure 15: Yamahata results of applying a 0.1 to 1.5 V load to the chevrons. ....	24
Figure 16: Yamahata results of applying a 0.9 to 1.1 V load to the chevrons. ....	25
Figure 17: Precise and accurate (top left), imprecise and accurate (top right), inaccurate and precise (bottom left), and inaccurate and imprecise (bottom right) results. ....	27

Figure 18: Two functions, one a phase shifted version of the other.....	28
Figure 19: The sum of products for each phase shift $\delta$ applied to Figure 18. ....	30
Figure 20: An ROI from a sample chip .....	34
Figure 21: Truncated sinusoid approximation of Figure 11, used for simulation purposes. .....	35
Figure 22: Same simulated function as in Figure 21, with a phase shifted equivalent in red. The shift in this example is four pixels. ....	35
Figure 23: The results of the cross correlation (above) and the region of the peak value (below). ....	36
Figure 24: The results of the cross correlation (above) and the region of the peak value ( $\sim 4.3$ pixels), with a 0.1 subpixel spline fit (below). ....	37
Figure 25: Plots showing different levels of sub-pixel upsampling, showing 1, 1/10, and 1/100 pixel samples. ....	38
Figure 26: The deviation of the spline fit as the actual displacement ranges from 4 to 5 pixels. Y axis units are pixels. ....	41
Figure 27: A zoom in on Figure 20, showing the noise in the image.....	42
Figure 28: A sample of noisy data based on Figure 27, with a maximum noise of 0.02 units. ....	43
Figure 29: Pixel deviation error for simulated images with a noise value of 0.003. Y axis units are pixels. ....	44
Figure 30: Pixel deviation error for simulated images with a noise value of 0.010. Y axis units are pixels. ....	44
Figure 31: Pixel deviation error from noise values of 0.001 to 0.010. ....	45
Figure 32: An offset simulation with an offset value of 0.1. ....	46
Figure 33: A higher contrast, and cropped version of Figure 20, showing how the left side of the image is lighter than the right. ....	48
Figure 34: A simulation of the slope result, based on Figure 33 with a slope of 0.10. ....	48
Figure 35: Figure 33 again. This time note that the brightness of the light regions varies more than the dark regions. ....	49

Figure 36: A simulated representation of an uneven illumination.....	50
Figure 37: Microscope and camera setup .....	52
Figure 38: The chip under the microscope, with two ribbon cables connected to the chip. .....	53
Figure 39: The camera attached to the microscope.....	54
Figure 40: MEMS chip used. ....	55
Figure 41: A zoom in showing the MEMS devices. ....	56
Figure 42: One of the devices on the MEMS chip. ....	57
Figure 43: User prompts in the XCorr software.....	58
Figure 44: Choosing the moving ROI.....	59
Figure 45: Choosing the reference ROI.....	59
Figure 46: Raw (above) and adjusted (below) data for the first an (red) and last (blue) images .....	61
Figure 47: Zoom in on Figure 46, showing the difference between the first (red) and last (blue) images. 163 nm pixel, ~2 nm motion. ....	61
Figure 48: Cross correlation of data from Figure 43 (above) and a zoom in on the peak value and 0.1 pixel spline fit (below). Y axes are arbitrary units. X axis units are pixels..	62
Figure 49: Stationary (Off) test: plot of voltage with respect to an arbitrary time .....	64
Figure 50: ROIs used in stationary test. Orange indicates moving ROI, and purple indicates reference ROI.....	65
Figure 51: Results of Test 5 (Off), showing the position measurements of 50 images for XCorr and Yamahata. ....	66
Figure 52: Standard deviation of measurements for the five tests. Average value for XCorr was 0.27 nm, and average value for Yamahata was 1.21 nm. ....	67
Figure 53: 1 Volt (On) test: plot of voltage with respect to an arbitrary time .....	68
Figure 54: 1 Volt (On) test: standard deviations $\sigma$ of five tests. Average $\sigma$ for XCorr was 0.36 nm, and average $\sigma$ for Yamahata was 1.23 nm. ....	69
Figure 55: 100 mv step test: Plot of voltage with respect to an arbitrary time. ....	71



Figure 56: The XCorr and Yamahata results of a 25 nm step. ....	72
Figure 57: Plot of voltage with respect to an arbitrary time. ....	73
Figure 58: XCorr and Yamahata results of a 2.5 nm step. ....	74
Figure 59: XCorr and Yamahata grouped averages for 2.5 nm step.....	75
Figure 60: 10 mv step test: XCorr results for a 2.5 nm step. ....	75
Figure 61: ROI (in orange) used in order to evaluate the noise in the data. ....	77
Figure 62: Adjusted intensity column averaged values of the ROI in Figure 61.....	78
Figure 63: A sample image (top left) and it's red, green, and blue components. ....	79
Figure 64: A sample Bayer filter, with the repeating pattern emphasized in the top left corner.....	80
Figure 65: Results of greyscale, green, red, and blue color channels (1 – 1.01V 2.5 m steps).....	81
Figure 66: The initial ROI (orange), and the one period ROI (purple, dashed line).....	83
Figure 67: ROIs used when evaluating impact of period number on measurements precision.....	84
Figure 68: The results for 1-6 periods of Yamahata and Xcorr.....	85
Figure 69: Zoom in of Figure 68, for the purposes of better visualizing the results. ....	85
Figure 70: ROI selected for the “half period” tests.....	87
Figure 71: The raw (above) and adjusted (below) column averages of the first (red) and last (blue) images used in an edge detection test (2 nm motion).....	87
Figure 72: XCorr 1/2 period ROI vs. Yamahata with a 1 period ROI. ....	88
Figure 73: A comparison of the original ROI, and the smallest tested when evaluating the impact of ROI. ....	89
Figure 74: ROIs used in determining impact of bar height in measurement precision. ..	89
Figure 75: The results of the tests for XCorr and Yamahata, as well as their power fits. ....	90
Figure 76: The moving ROI (orange) and the reference ROI (purple) for the dimples. The previously used ROI on the bars is also indicated (green).....	92

Figure 77: Column averages of the bars (above in green) and dimples (below in orange).  
..... 93

Figure 78: An irregular structure used in testing. The moving ROI (orange) and reference ROI (purple) are indicated..... 95

Figure 79: The irregular structure when converted to numeric values. Moving ROI above (in orange), reference ROI below (in purple)..... 96

## ABSTRACT

This thesis examines the development and testing of a new algorithm for measuring rectilinear displacements in Micro Electro Mechanical Systems (MEMS), however the algorithm could easily be applied to other applications. This optical measurement technique is based on cross correlation used in signal matching. The primary objective of this thesis is to develop a system with improved resolution and versatility which can be applied to a wide variety of MEMS chip designs. Experiments testing this new algorithm had a standard deviation in the measurement as low as 0.23 nm (equivalent to less than 1/700 of a pixel). This result is approximately 4.5 times better than previously found in the Dalhousie MEMS lab using existing algorithms. In addition, when the conditions of the measurement were made worse, the new algorithm consistently outperformed the existing algorithm. Further development may be able to yield additional improvements in the measurements.

## List of Abbreviations of Symbols Used

1D	1-dimensional
2D	2-dimensional
C	Cross Correlation
CSV	Comma Separated Variables
DIC	Digital Image Correlation
FIB	Focused Ion Beam
FFT	Fast Fourier Transform
IMUDTMCI	Chip code of MEMS chip used
$k_m$	thermal conductivity
$L$	Length
MEMS	Microelectromechanical Systems
MIG	Median Intensity Gradient
MUMPS	Multi User MEMS Process
$P$	perimeter
PolyMUMPS	Polysilicon based MEMS process
Pt	Platinum
PZT	Lead Zirconate Titanate
$R$	Resistance
$R^2$	Coefficient of determination
RGB	Red, Green, and Blue colour values
RIE	Reactive Ion Etching
ROI	Region of Interest
SSSIG	Sum of Square of Subset Intensity Gradient
uint	unsigned integer
VI	Virtual Instrument
XCorr	Cross Correlation optical measurement program
$x_0$	delay factor

$\alpha$	coefficient of thermal expansion
$\Delta z$	vertical gap
$\delta$	discrete pixel shift
$\lambda$	wavelength
$\sigma$	error, standard deviation
$\theta$	angle
$\nabla$	gradient

# 1. Introduction

## 1.1. Objective

The objective of this thesis is to develop an improved algorithm for measuring  $\mu\text{m}$  and  $\text{nm}$  scale 1-D displacements in Micro Electro Mechanical System (MEMS) applications. The thesis will provide a brief overview of the MEMS devices used, the theory behind the newly developed measurement algorithm, and how it compares to existing algorithms.

MEMS devices often use actuators for manipulation, and most of these are one dimensional. This typically means that displacements occur in one direction, which has led to the development of a variety of measurement algorithms to produce highly precise measurements which are optimized for the simple motions observed. The success of the newly developed algorithm will be evaluated in three key ways:

1. What is the expected precision of a measurement under the ideal conditions?  
This will be measured by determining the standard deviation of measurements taken while a system is stationary.
2. What is the smallest displacement that the system can detect? This will be evaluated by applying progressively smaller steps to a device and taking multiple measurements of each of these displacements. For the purpose of this thesis if the standard deviation of a measurement is less than or equal to one quarter of the measured step, then the step is considered detected.
3. How does the precision degrade as the conditions of the test deteriorate? This will also be measured by determining the standard deviation of measurements taken while a system is stationary, and then changing elements of the system to determine how much, on a percentage basis, the measurement changes.

Based on previous work in the literature, the standard deviation of measurements is the most common metric used to evaluate the precision of a measurement algorithm. Previous papers use 1, 2, or 4 standard deviations, however for

the purpose of consistency in this thesis, they will all be converted to 1 standard deviation.

## 1.2. Scope and Limitations

This thesis will present the theory, as well as develop the software for performing sub-nm MEMS measurements. The software for measuring the displacement in the images was developed in MATLAB, and a series of LabVIEW VIs were developed by modifying some which had previously been created in the MEMS Lab in order to run the necessary tests.

The measurement algorithm is limited to measuring rectilinear displacements in one dimension, though it could in theory be used for two-dimensional motion assuming there is no rotation occurring. This would require either modifying the software to select for both directions, or running the analysis once for each desired direction.

## 2. Literature Review

### 2.1. MEMS Devices

MEMS devices are fabricated devices which can provide motion or sensing on the microscopic scale. There are a variety of techniques and processes used for the fabrication of MEMS devices [1], with typical structure features having dimensions of a few microns, as well as some sub-micron sized devices [2]. The devices in this thesis use actuators controlled by a current running through them and produce motions ranging from nanometers to microns [3]. Due to the scale in which these structures exist, forces that scale with area and length (such as adhesion) tend to be much more prominent than those which scale with volume (such as weight) [4, 5].

### 2.2. Optical Displacement Measurements

#### 2.2.1. Why are Optical Measurements Used?

Optical measurement algorithms are frequently used in MEMS. Data from MEMS chips is often collected in the form of images, either photographic still images or as part of a video. By comparing images via computer algorithms, it becomes possible to produce extremely precise measurement results.

The images taken by the computer do have limits to their resolution, regardless of magnification present in the microscope. Using the Rayleigh criterion, the resolution limit is generally assumed to be:

$$Resolution = \lambda/2 \quad (1)$$

The wavelength,  $\lambda$ , of the visible spectrum of light ranges from about 400 nm to 700 nm, and so in an ideal case in which only blue light is present in a system, there is a limitation of roughly 200 nm resolution (slightly larger than the 163 nm pixel size which is used). Two small specs of dust, for example, that were within 200 nm of one another could not be distinguished through the microscope regardless of the pixel size of the camera. This resolution of approximately 200 nm is much larger than the displacements



that are desired to be measured. However, this does not present a problem in this application because the 200 nm resolution applies to distinguishing multiple distinct objects.

The Rayleigh criteria does not apply to the measurement of motion of objects, only the resolving of objects. The Rayleigh criteria evaluates whether or not individual objects can be distinguished from one another, however it does not define how precisely the position of an object can be determined.

### 2.2.2. Digital Image Correlation

In Civil and Mechanical Engineering applications, optical measurements are often used to determine strains in a system. Digital Image Correlation (DIC) refers to techniques which can evaluate the motion in a plane from a series of images (see Pan 2009 [9]). An image of a stressed component is compared to an original reference image and changes between these are measured to calculate a 2D strain field. All DIC systems are capable of sub-pixel resolution (see Pan 2010 [6]).

Commercial DIC algorithms are optimized to measure translation, rotation, and deformation mapping where combined axial and shear stress distort the image [7]. DIC systems handle an unknown combination of translation, rotation and distortion and typically use iterative methods (e.g. Newton–Raphson [8]) to converge on the solution. DIC algorithms look for changes in patterns in the image. If no clear patterns on a structure exist, patterns can be added to the structure prior to imaging. Random speckle patterns are typically used in order to measure and extract image translation, rotation and distortion [9, 10]. Speckle patterns consist of a series of small markings on a surface. Typically the markings are randomly distributed across a surface and are roughly circular, though shapes and size may vary.

Generally, large deformations and rotations have the largest impact on the resolution DIC, both will cause notable reductions in the quality of the subpixel accuracy [6]. As DIC algorithms measure sub pixel deformations, other smaller factors which would not normally be an issue can decrease the achievable resolution. The details of

the speckle pattern such as density, contrast, and fill factor can all effect the result and DIC packages may apply filters to the geometry of images in order to simplify the data [11]. Camera effects such as: changes in illumination, out of plane motion, non-parallel image and camera planes, and camera lens distortion can also degrade the results (see Pan 2009 [12]). The correlation interpolation method can also introduce errors in the results.

A sample of a speckle pattern is shown in Figure 1. In the image to the left, a small region is identified with four dots in a diamond shape. In the following image, after translation, rotation, and deformation have occurred, the same diamond can be found again. A DIC algorithm will identify this region, and evaluate the translation, rotation and deformation which caused it.

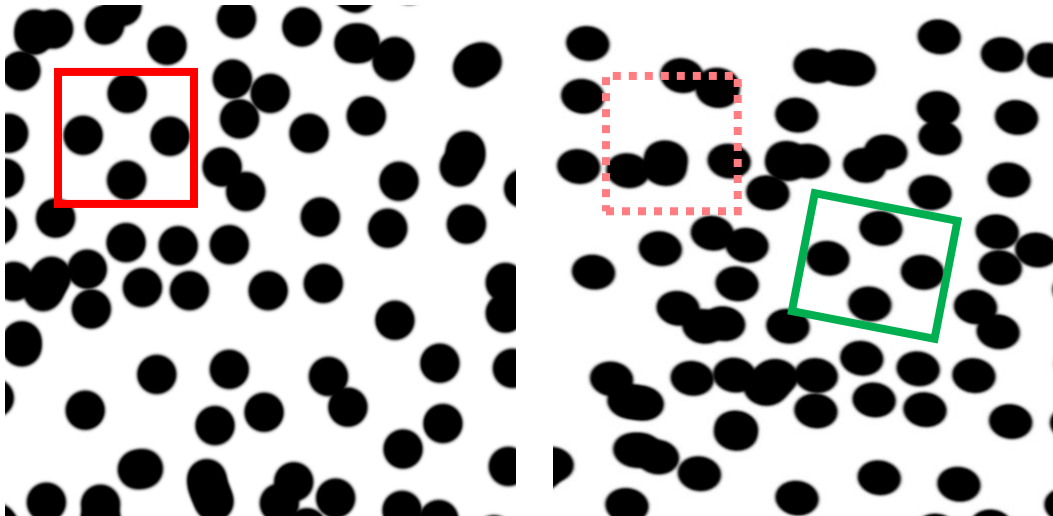


Figure 1: A representation of a speckle pattern which would be tracked by a traditional DIC algorithm.

MEMS DIC using a microscope setup has a number of advantages over conventional macro scale DIC:

1. The camera axis is perpendicular to the image plane with little or no change in angle.
2. In MEMS there is typically little to no out of plane motion.
3. Microscope optics are usually of high quality, reducing lens distortion effects.
4. The sample illumination can be tightly controlled.

MEMS DIC does have a number of disadvantages:

1. The optical limits of resolution and diffraction effects may be an issue.
2. The surface contrast is set by the materials used and not easily changed.
3. Applying micron or sub-micron scale speckle patterns may not be possible.

As a result, the motions being evaluated in the MEMS devices being studied for this thesis will behave as shown in Figure 2. The displaced image on the right only experiences translation, and only in one direction. As this is known, the algorithms can be designed to accommodate this, and will be made more efficient and more precise with fewer variables to consider.

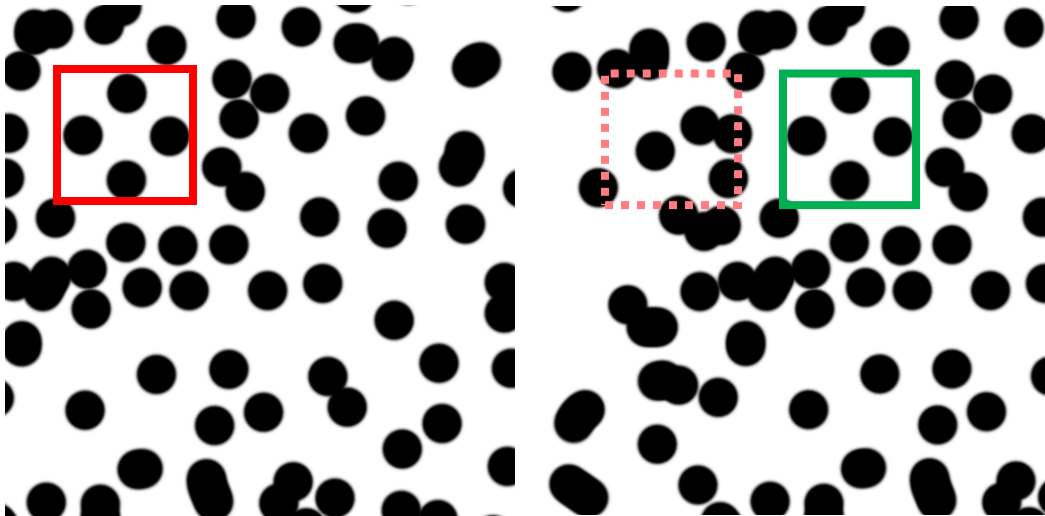


Figure 2: A representation of a speckle pattern which would be tracked on a MEMS device.

### 2.2.3. MEMS Optical Measurements

In MEMS, DIC algorithms similar to those used on the macro scale have been used to measure such quantities as stress and Young's modulus. For example Yagnamurthy [13] calculated full field strains on PZT films using deposited speckle patterns. Robin [14] measured stress strain curves for SU-8 films speckle patterned using nano-powdered copper oxide particles. Roland [15] used DIC to measure tensile stress on thin Au and Pt films, where microstructural features of the films served as

natural speckle pattern. Using  $0.1 \mu\text{m}$  pixels and a  $150 \times 115 \mu\text{m}^2$  region they report uncertainties of  $1/250$  of a pixel ( $\sim 0.4 \text{ nm}$ ).

While DIC techniques have frequently been used to measure MEMS strain deformation and MEMS material properties, they are less commonly used to measure MEMS displacements (as opposed to strain fields). One issue is the need for speckle patterns, although this has been overcome. For example, Berfield [16] measured rigid body displacement and performed tensile tests of transparent polymers coated with fluorescent nano-particles (140-180 nm mean diam). They report a noise level of  $\pm 7 \text{ nm}$  with a  $0.213 \mu\text{m}$  /pixel ( $\sim 1/30$  of a pixel). As another example Naraghi [17] measured the displacement of an electrostatic MEMS actuator to an accuracy of  $20 \text{ nm}$  (no pixel size reported). In this case micron and sub-micron scale circular pits generated using FIB (Focused Ion Beam) were used as the speckle pattern.

In MEMS, many displacement measurements are rectilinear, and as such the complexity of the two dimensional DIC algorithms is not always required. By using algorithms specifically designed for the one dimensional problem, it is possible to improve the accuracy and precision of the results, while also reducing computation time. There are a number of existing techniques used in MEMS for evaluating rectilinear motion:

#### 1) Edge methods

The first technique to be discussed was developed by Ya'akovitz [18] for motion detection of an electrostatically actuated MEMS device. The technique uses DIC methods on regions with a sharp change in contrast, such as an edge, to evaluate displacements. The algorithm separates regions based on intensity in a greyscaled image and evaluates the motions of these regions. Using a pixel size of  $400 \text{ nm}$ , Ya'akovitz reports resolutions of  $16 \text{ nm}$  ( $1/25$  of a pixel).

#### 2) Moiré Methods

A Moiré pattern uses a series or pattern of fine lines. By downsampling (skipping over pixel data in even intervals) a beat pattern is formed between the sampling period

and the fine pattern. This beat pattern is larger than the fine period and varies with the phase of the fine pattern. This can be measured and compared across multiple images in order to determine displacement. Sugiura [19] has developed a Moiré algorithm and has shown results with a standard deviation of approximately 14 nm, (no pixel size reported). Chang [20] has also developed a Moiré algorithm using smaller periodic structures and a similar method to that of Sugiura; a standard deviation of approximately 1.6 nm was obtained (no pixel size reported).

### 3) Fast Fourier Transforms (FFT) methods

Yamahata [21] has developed an algorithm which uses FFT in order to determine the phase shift between images. A section of the image is column averaged to create a 1D profile. An FFT is performed on the profile and the phase calculated. The phase difference from image to image is multiplied by the spatial wavelength of the profile yields the motion. The algorithm will be explored in more detail in Chapter 2.3. Using a pixel size of 75 nm, Yamahata reports resolutions of 0.13 nm for one standard deviation (1/500 of a pixel). This was when using a column averaging height of 150 pixels. The Yamahata algorithm was implemented in the Dalhousie MEMS lab, but the above level of precision was not achieved. Previous experiments at Dalhousie using a 163 nm pixel size had shown a precision of approximately 2 nm (1/80 of a pixel). This has since been improved by the author to approximately 1.25 nm (1/140 of a pixel) when using a column height of 250 pixels.

Cheng et al [22] used a different FFT image algorithm which was used to measure micro-cantilever beams. FFTs of a reference image and of target images were computed. Highly precise results were yielded by using a least squares method. Using a pixel size of 130 nm, they reported a noise level of 0.10 nm (1/1300 of a pixel).

### 4) Curve fitting algorithms

Kokorian [23, 24] developed an algorithm uses a smoothed spline fit of image data. A section of the image is chosen that has a fixed edge and a moving edge and the image column averaged to create a 1D profile with two edges/peaks. The profile is modeled using a spline, with a shift or delay factor  $x_0$ , which is used to represent a phase

shift between the two edges/peaks. An algorithm then will seek out the optimal value of  $x_0$  for each new image. Using a pixel size of 32 nm, Kokorian reported a standard deviation of 0.06 nm (1/500 of a pixel). This was when using a column height of 1577 pixels. Unlike the Yamahata algorithm, the Kokorian algorithm does not require periodic structures.

Table 1 compares the results of several of the discussed displacement measurement methods, in terms of pixel size, precision, and resolution.

Table 1: Comparison of several displacement measurement methods

<b>Author</b>	<b>Method</b>	<b>Pixel Size (nm)</b>	<b>Precision (nm)</b>	<b>Resolution (pixels)</b>
Berfield	DIC	213	7	1/30
Roland	DIC	100	0.4	1/250
Ya'akobovitz	Edge Detect	400	16	1/25
Sugiura	Moire	-	14	-
Chang	Moire	-	1.3	-
Yamahata	FFT	75	0.13	1/500
Dalhousie (Yamahata)	FFT	163	1.2	1/130
Cheng	FFT	130	0.10	1/1300
Kokorian	Spline Fitting	32	0.06	1/500

### 3. Existing MEMS Devices

#### 3.1. PolyMUMPs

The process used for manufacturing the MEMS devices for Dalhousie University is a variation of the Multi User MEMS Process (MUMPS). The specific variation is called PolyMUMPS, so named because of the three layers of polysilicon which are used to make up many features on the chips [25].

Figure 3 shows a visual representation of the layers used in the PolyMUMPS process. Starting at the bottom, the nitride layer is an insulating base on which the other materials are deposited and typically covers the entire chip. The oxide layers are temporary support layers which are removed via an etching process. The most important layers are the Poly layers, which are used to form the structural components. These layers are made of the previously mentioned polysilicon, a polycrystalline form of silicon with a high purity. In contrast with monocrystalline structures, which have a homogenous, unbroken crystal structure, polycrystalline structures contain multiple different crystal structures. This can produce a visible grain structure within the silicon. Polysilicon has historically been used in the electronics industry, including use in photovoltaic cells [26, 27]. They fulfill a similar role to that of support material in rapid prototyping. The metal layer is generally used as a means of carrying current.

The dashed line on the first oxide occurs  $0.75\ \mu\text{m}$  from the top of the layer and is used for applying “dimple” features. These are small holes in the oxide layer which will produce small indents on the Poly 1 layer. These dimples are required in many devices, as they will significantly reduce friction on a component. Flat surfaces are prone to sticking together in MEMS devices due to friction, and the dimples will prevent this by significantly reducing the contact area between two faces.

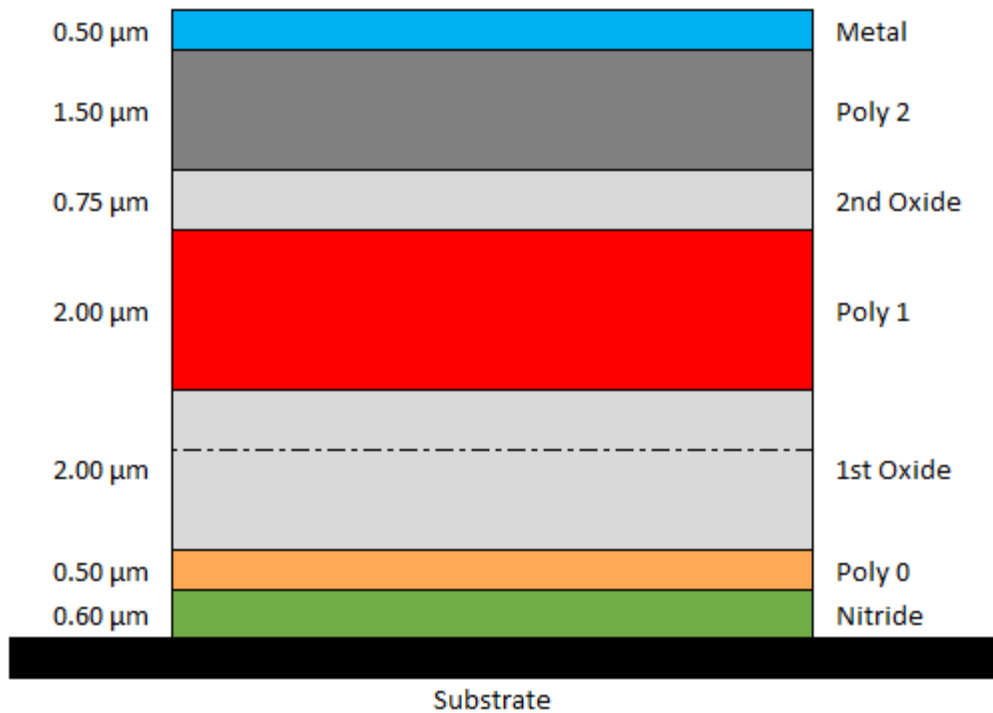


Figure 3: PolyMUMPS layers

The production of MEMS devices in this process works by applying one uniform layer at a time, then applying a photoresist to the layer. Light will be applied to the photoresist in the desired pattern. The photoresist material which is exposed to light can be chemically removed from the chip, leaving a patterned photoresist on the chip. Reactive Ion Etching (RIE) is used to remove any material in the current layer not protected by the photoresist (see PolyMUMPS Design Handbook [25]). After this, the photoresist is removed from the chip, and the next layer is applied. This process is repeated for each of the following layers.

After all the layers have been deposited on the chip, the two oxide layers are removed via the use of wet etching: a bath containing hydrofluoric acid is used for this removal. This will remove both oxide layers but leave behind the nitride, up to three poly layers, and the metal layer.

Figure 4 shows a simple device, a cantilever beam, as it would be created on a PolyMUMPS chip. In this case the orange layer is Poly 0, and the red layer is Poly 1. Between the overhanging Poly 1 and the Poly 0 an oxide layer would have been used



and been removed via etching. No oxide layer was present at the anchoring point on the left side, allowing the Poly 1 to bond directly to the Poly 0.

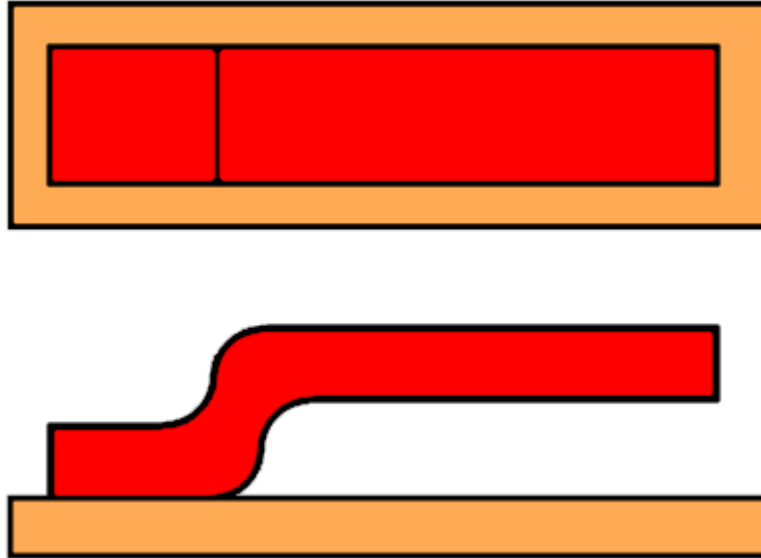


Figure 4: A simple cantilever beam composed of Poly 0 (orange) and Poly 1 (red) layers in a PolyMUMPS. MEMS device, as seen from the top (above) and side (below).

### 3.2. Thermal Actuators

To test the cross correlation algorithm's ability to measure motion in a MEMS device, a method for moving objects on the MEMS chip is required; chevron thermal actuators are used. Figure 5 shows a schematic of a chevron actuator as it is designed in the software package L-Edit. In these devices, two sets of slender beams are anchored at one end and meet at the middle. The beams are at a small angle normal to the motion (in this case about  $6^\circ = 0.1$  rads). The beams themselves consist of  $140\ \mu\text{m}$  long Poly 1 arms  $2 \times 2\ \mu\text{m}$  in cross section. The beams are elevated  $2\ \mu\text{m}$  above the substrate. A voltage of a few volts is applied across the anchored ends, causing a current of a few milliamps to flow through the beam. This current will cause heating to occur in the thin beams, causing thermal expansion. The beams push against each other and the bent angle of the beams forces them to move in the perpendicular direction.

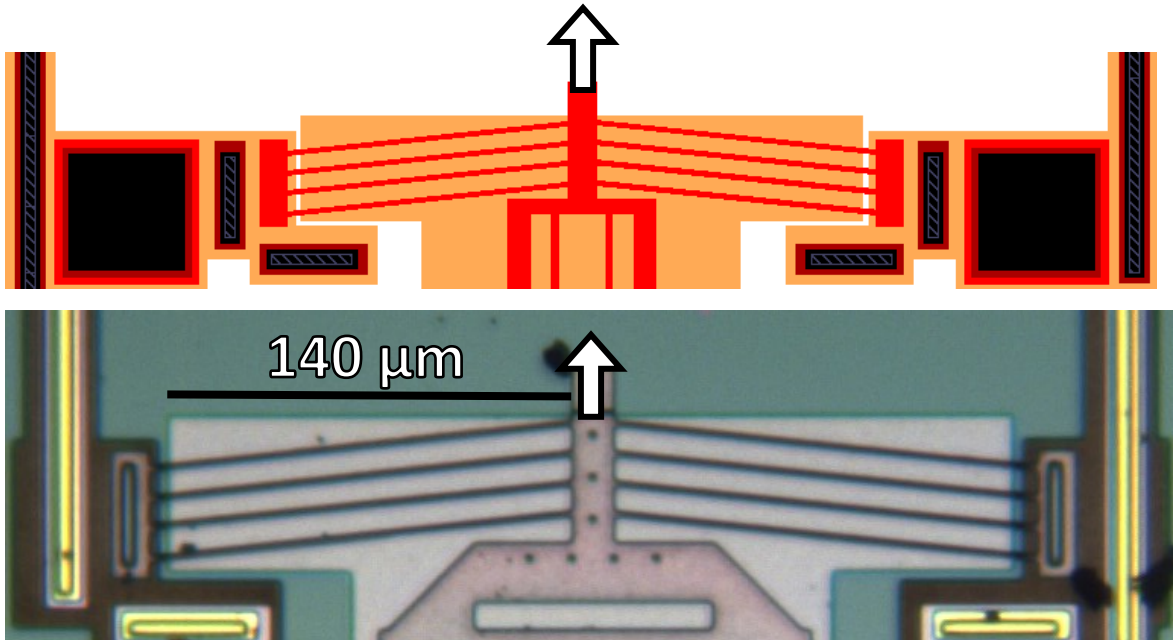


Figure 5: L-Edit model of a chevron actuator (above) and photo of an actuator on MEMS chip (below). The direction of motion is indicated.

The existing thermal actuator design is effective at generating easily controlled and repeatable motion in a variety of MEMS devices, however the amount of motion which can be generated is limited to a few microns. When the voltage is increased too much the heat generated in the thermal actuator will cause permanent deformation or fracture.

Using the same concepts, further amplification can be generated by using two linear actuators. A sample L-Edit model, as well as the component on a chip are shown in Figure 6. These actuators are set up such that their linear motion will act along the same line, but in opposite directions. The motion will then be applied to additional polysilicon bars at the same initial angle as the linear actuators. The output will be amplified a second time, allowing for roughly 10 times the range of motion.

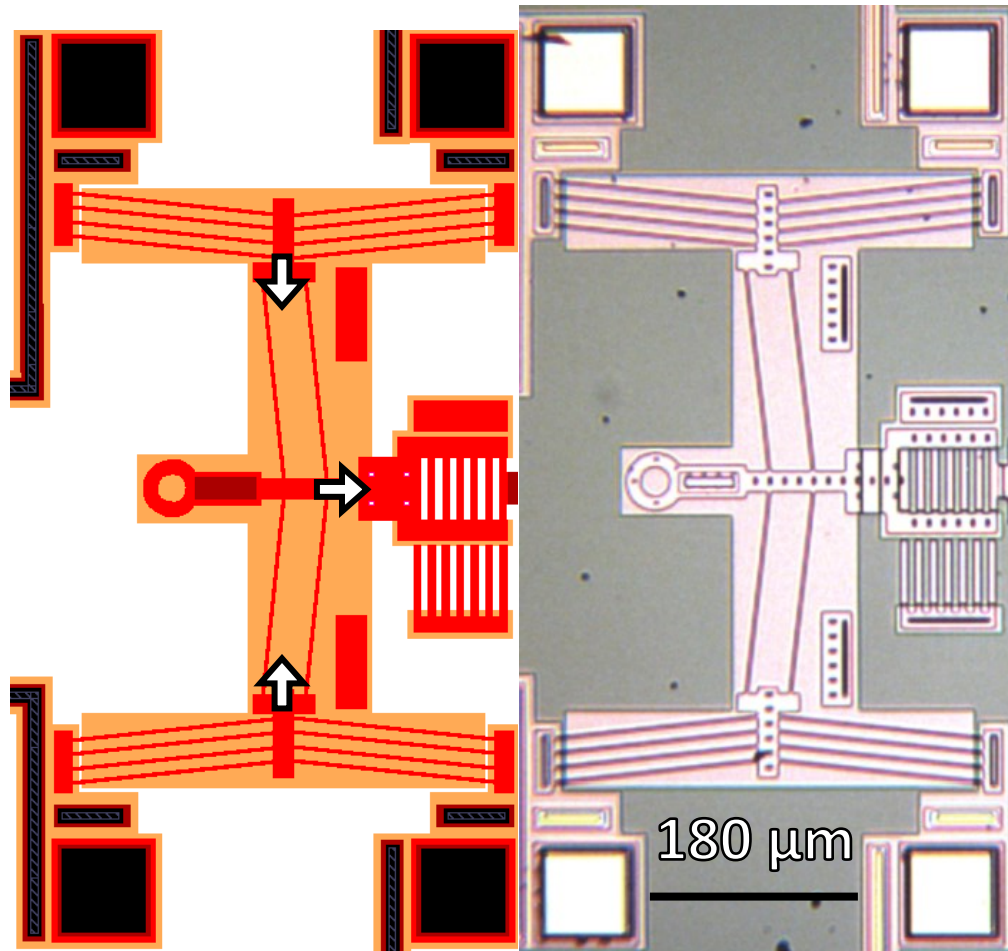


Figure 6: A double thermal actuator design in L-Edit (left) and on the MEMS chip (right).

The thermal actuators used to manipulate the MEMS devices are controlled by running a current through the actuators. This current will cause an increase in temperature according to the following equation [28]:

$$\Delta T \approx \frac{2V^2 \Delta z}{Rk_m PL} \quad (13)$$

where,  $L$  is the beam length, equal to  $150 \mu\text{m}$ ,  $\Delta z$  is the vertical gap, equal to  $2 \mu\text{m}$ ,  $P$  is the perimeter, equal to  $8 \mu\text{m}$ ,  $R$  is the electrical resistance of the system, equal to  $1.3 \text{ k}\Omega$ , and  $k_m$  is the thermal conductivity of silicon, which is  $0.05 \text{ W}/(\text{mK})$ . The displacement is determined by the following equation:

$$\Delta y \approx \frac{L\alpha\Delta T}{\theta} \quad (14)$$

Where  $\alpha$  is the thermal expansion coefficient of silicon ( $2.6 \cdot 10^{-6} \text{C}^{-1}$ ) and  $\theta$  is the angle of the beams, in this case 6 degrees (0.1 rad) By bringing in the equation for  $\Delta T$ , the length will cancel out, and the equation for displacement will be:

$$\Delta y \approx \frac{2\alpha V^2 \Delta z}{\theta R k_m P} \quad (15)$$

This shows that the change in displacement should be proportional to the square of the voltage, assuming all other factors are held constant.

## 4. Existing Yamahata Algorithm

The Dalhousie MEMS lab has used the Yamahata algorithm for several years [29, 30], and it will serve as a baseline for comparisons of the newly developed algorithm. Before measurements can be taken, the image needs to be processed in a few ways.

### 4.1. Image Processing

The first change that is made to the images is converting them from RGB images to greyscale. This is done in order to reduce each pixel to a single value. In an 8-bit RGB image, each pixel has a value ranging from 0 to 255 for the red, the green, and the blue components. In these cases, 0 represents no presence of the colour while 255 represents the colour at full value. Once the image is converted to greyscale, each pixel is represented by a single value from 0 to 255. In this case, 0 would be a pure black and 255 would be a pure white. There are a number of different ways to make this conversion. The algorithm which is used in this work is a standard RGB to grayscale conversion:

$$X = 0.299 R + 0.587 G + 0.114 B \quad (2)$$

Where X is the output value for a given pixel, and R, G, and B are the 8-bit values for red, green, and blue respectively. There are many similar equations with variations in the coefficients for each colour, however changing from one to another did not have a notable effect on the results. Normally the value of X would be an integer, 8-bit value (UINT). However, to avoid rounding errors associated with integer values, the input image UINT values are first converted to doubles. Figure 7 shows the conversion of a colour image to a greyscale image.

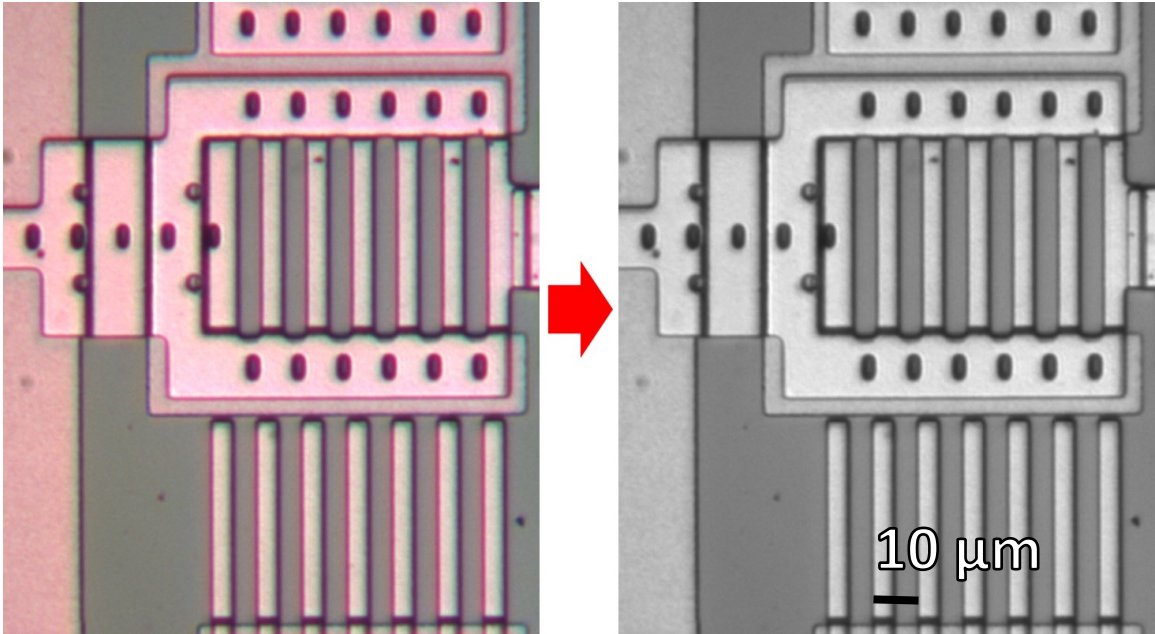


Figure 7: RGB image reformatted to greyscale.

Periodic structures are required by the Yamahata algorithm in order for the FFT obtain precise results, and this requires planning in the design of MEMS structures. This requirement means that the Yamahata algorithm is limited by to MEMS chip with periodic structures. This has led to a number of MEMS chips in the Dalhousie MEMS lab being developed with the “bars” and “dots” seen in Figure 7.

The next step in the image processing is the selection of a “region of interest” which contains only a periodic structure. These regions, typically referred to as ROIs, are the portions of the image which will be used for calculating motion. The bars and dots provide a consistent intensity within each column of the ROI. Figure 8 shows a MEMS device with an ROI selected by the red box, and a zoom in on the ROI.

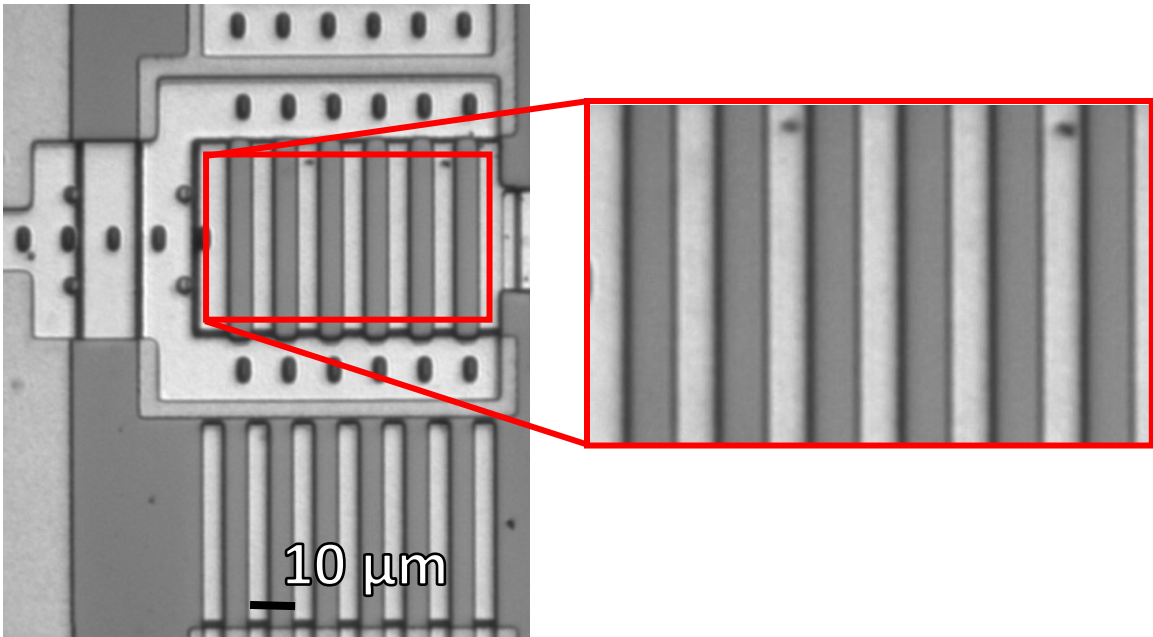


Figure 8: A greyscale image of a MEMS chip (left), and a region of interest used in measurements (right).

The ROI is then column averaged. This is done to reduce the noise of the system, and to convert the entire system into a 1D function of  $x$ . Figure 9 shows the 1D function of the ROI from Figure 8. The average value of the function (red line) is first subtracted from the entire function. From the entire data set (black and blue dots) the zero crossing points (green dots) are determined and a subset of an integer number of wavelengths is extracted (the blue dots). The Yamahata FFT is more accurate when analyzing an integer number of periods.

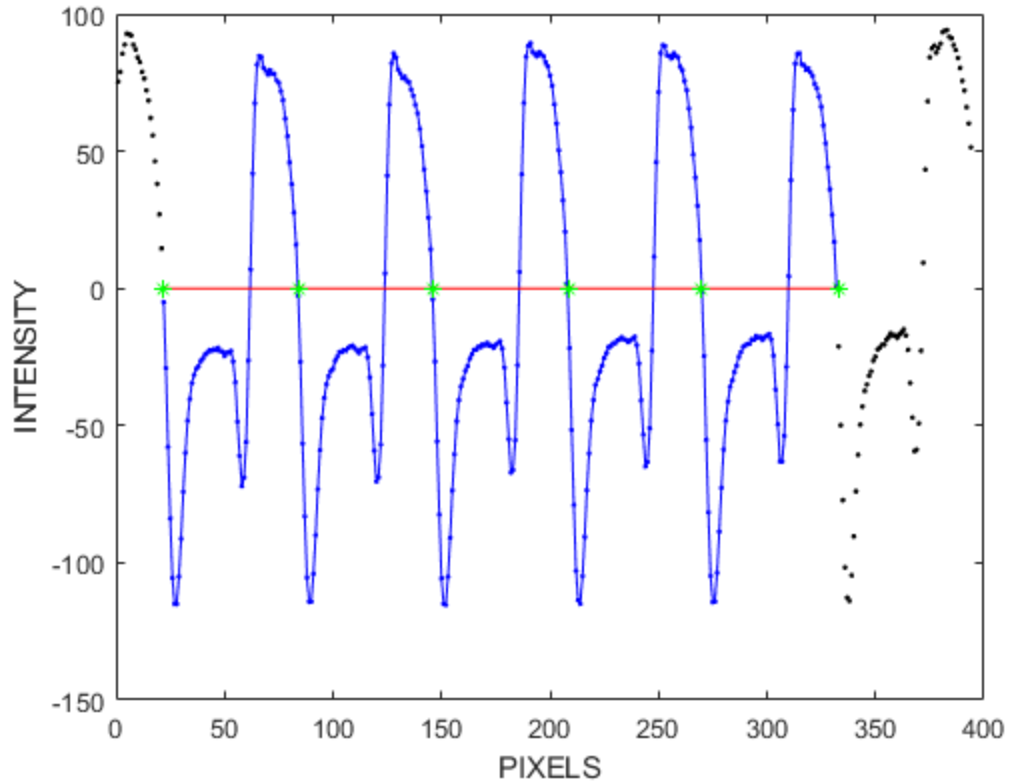


Figure 9: The column averaged values, showing 6 zero crossings (green) and 5 periods (blue) which will be used in calculations.

An FFT is taken of this 1D profile, and the magnitude is normalized on a second plot, shown in Figure 10. The peak value of this plot is the known spatial wavelength of the structure. In the case of these tests, most structures had a period of 10  $\mu\text{m}$ . This wavelength calculation is only computed for the first image, as all subsequent images are assumed to have the same spatial wavelength. The phase of the fundamental wavelength of each image is extracted from the FFT. The phase shift is calculated by subtracting the initial image phase from the phase of each other image. The phase shifts are then multiplied by the known spatial wavelength. The primary limiting factor of the FFT technique proposed by Yamahata is the requirement for periodic structures. Reducing the number of periods reduces the precision of the measurements, which limits the application of the algorithm (this will be further discussed in Chapter 6.3).



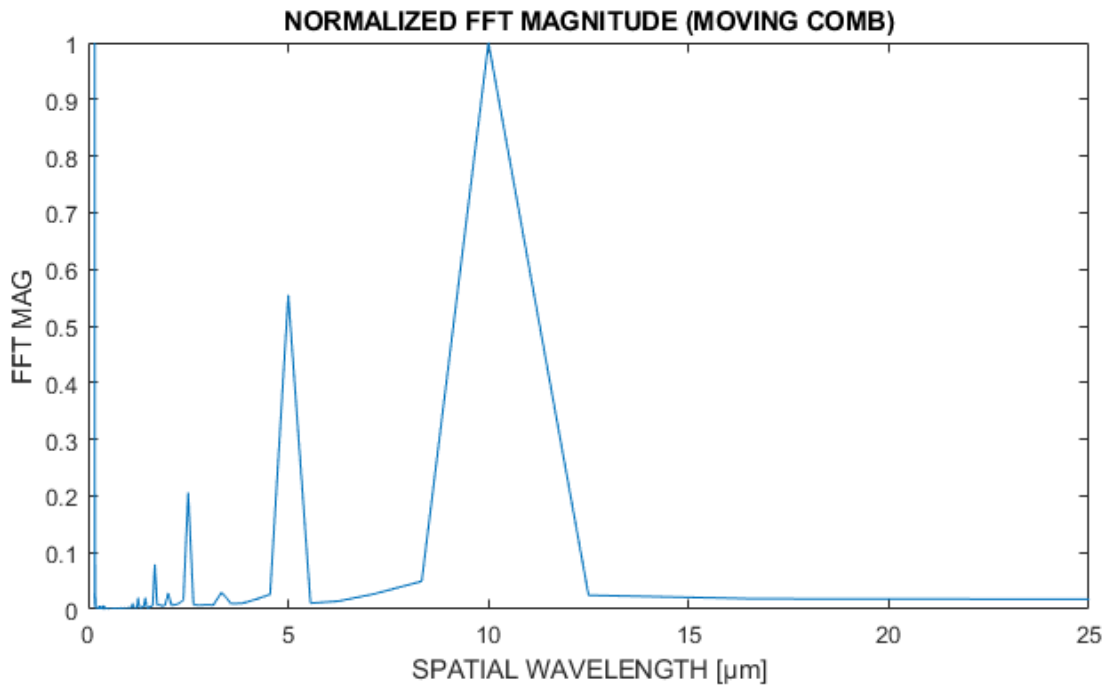


Figure 10: Normalized FFT of a 1D profile indicating a 10  $\mu\text{m}$  spatial wavelength.

#### 4.2. Reference Structures

The camera images will not remain perfectly stationary. Very slight motions can cause relative motions between the camera and chip. Motions may be caused by vibrations in the surrounding environment (these are reduced by an isolation table), slow drift of the microscope stage, and the camera not being perfectly rigid. This relative motion can be the dominant source of noise in measurements.

In order to counteract these motions, two ROIs are used. A moving ROI is chosen by using a periodic structure on the object of motion. A second ROI is chosen using a periodic structure that is intended to be fixed. As the reference is assumed to be completely stationary, any motion that is seen in the reference is assumed to have occurred through the entire chip. Figure 11 shows the same component from Figure 8, and identifies a pair of ROIs, with the moving ROI in orange, and the reference ROI in purple.

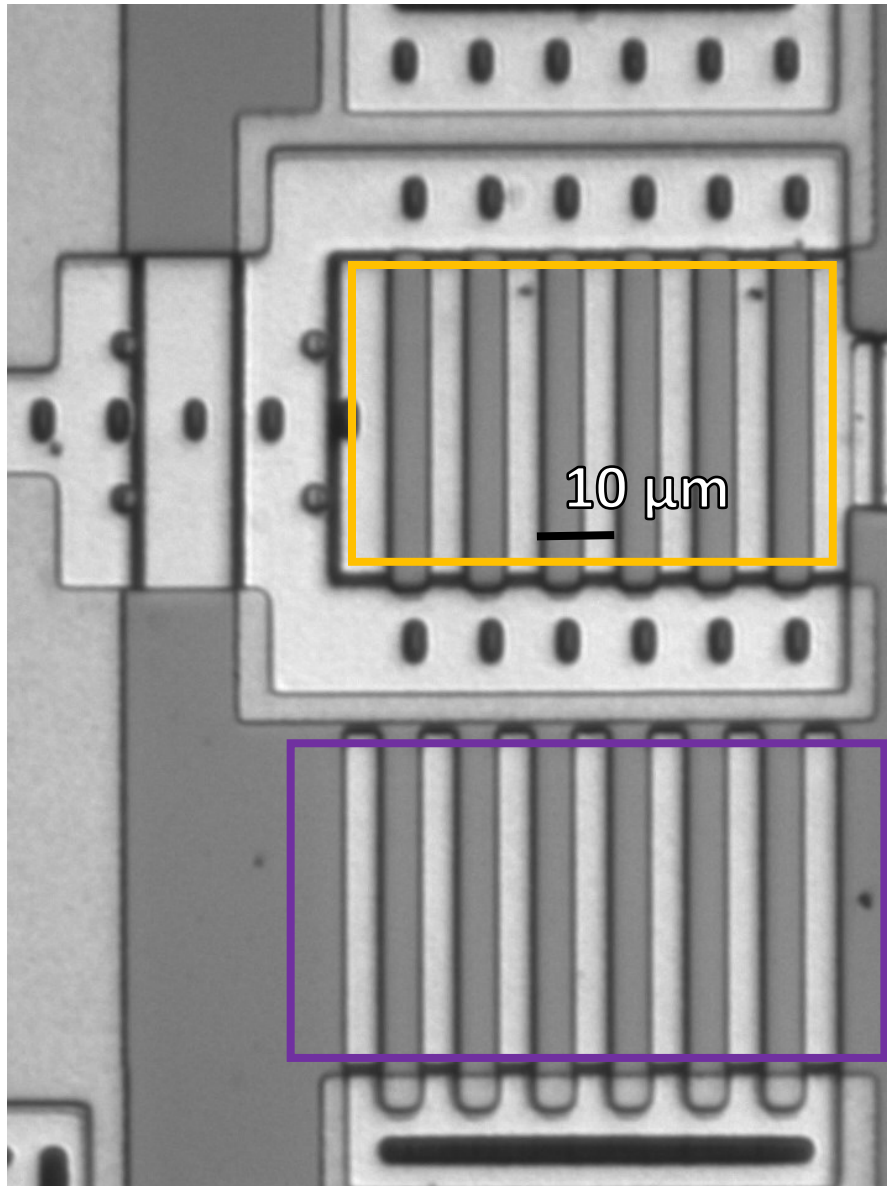


Figure 11: Component on a MEMS chip, with moving ROI (orange), and reference ROI (purple).

By taking the difference between the moving ROI and the reference ROI, the actual displacement of the moving components can be calculated. Figure 12 shows an example of these results. In this case, the “moving” ROI is intended to remain completely stationary. However, without the reference to correct for motion in the chip, the image experiences relatively large, random displacements of about 20-30 nm.

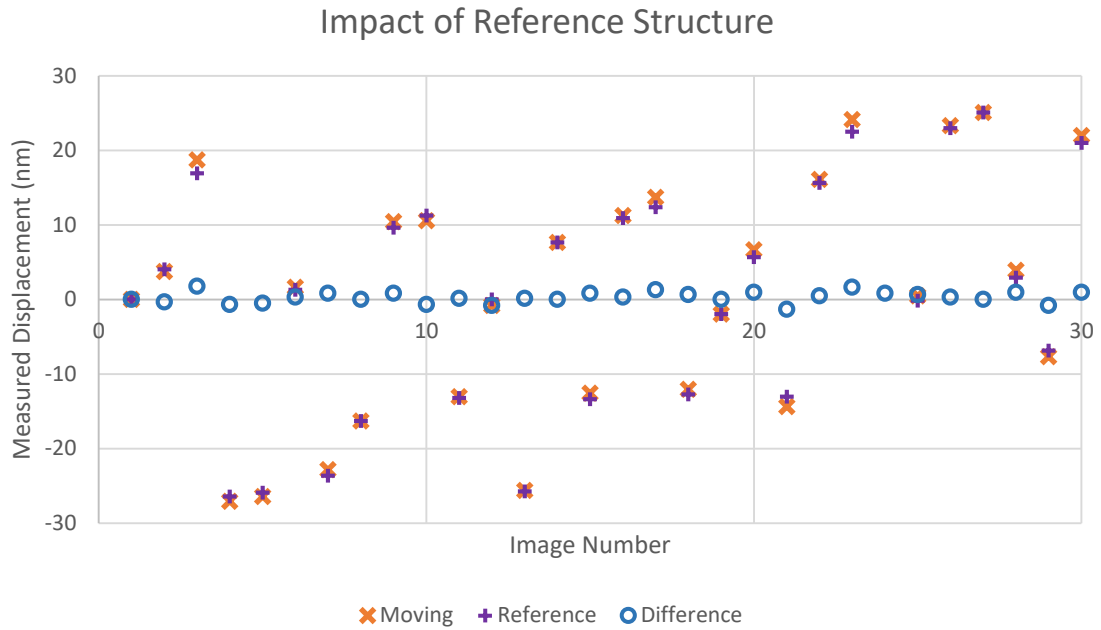


Figure 12: Measurements of moving and reference structures, as well as their difference.

#### 4.3. Typical Chevron Motion

A sample of motion from a chevron actuator can be calculated using the Yamahata algorithm. Based on the equations presented above, it is expected that the motion should vary parabolically with voltage. Figure 13 shows the actuator on the left, as well a zoom in on the right.

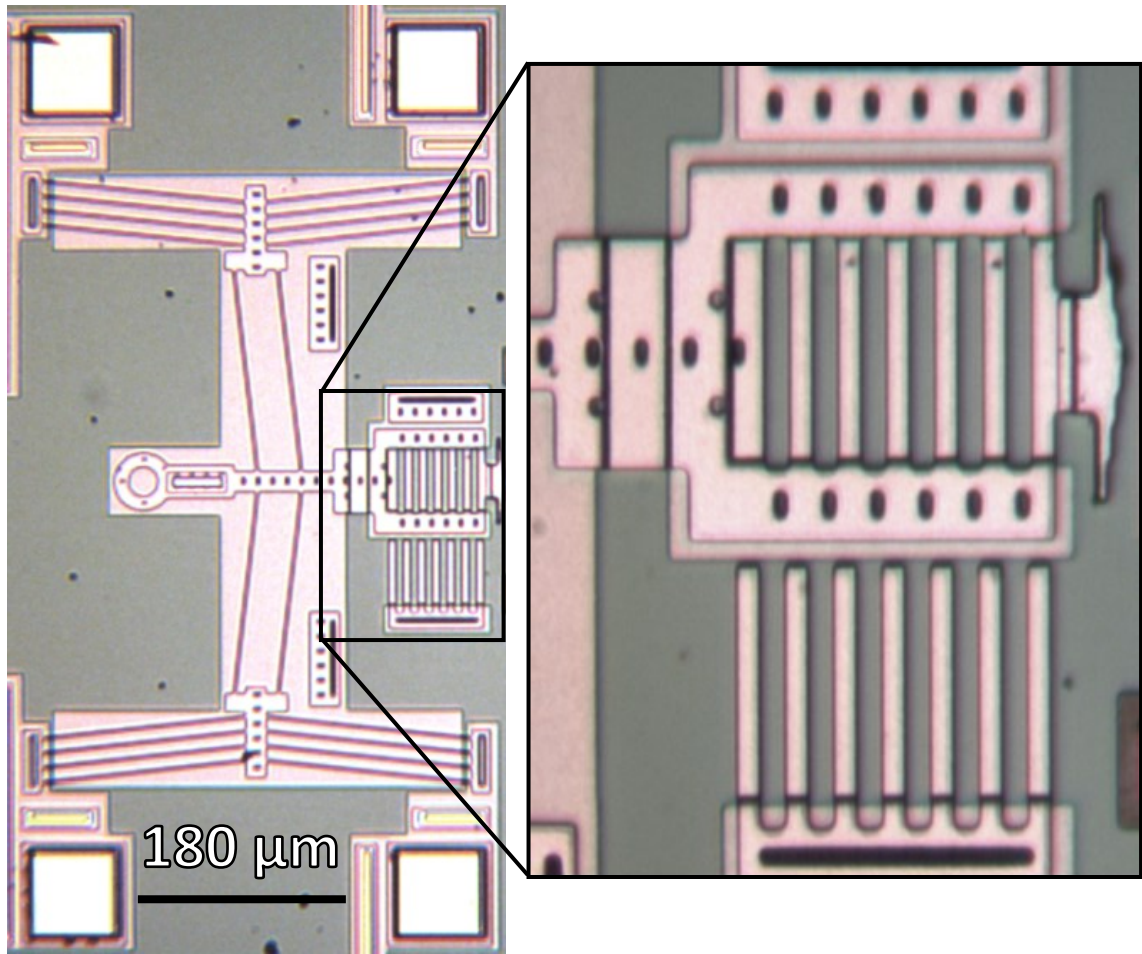


Figure 13: A MEMS device using chevron actuators (left) and a zoom in of the device (right).

The result of zooming in, choosing an ROI and processing the image, is shown in Figure 14. As can be seen, there is a notable level of noise from one pixel to the next. There is also, near the top of one of the rightmost bars, a small speck of dust or dirt.

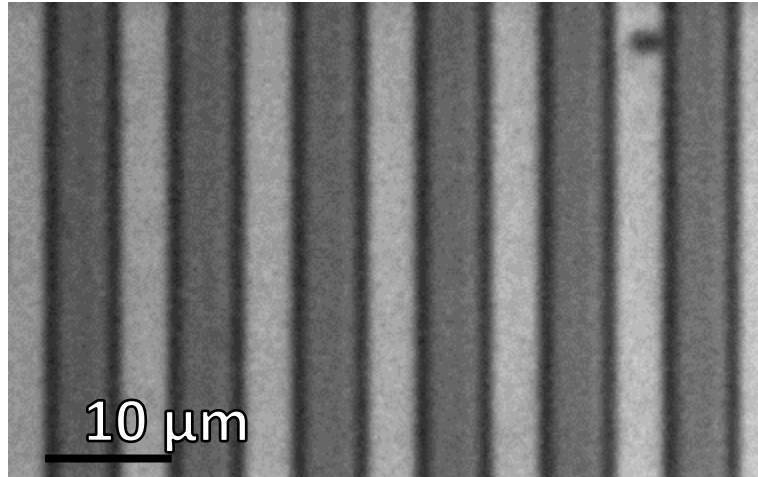


Figure 14: The ROI used to determine the motion created by the actuator.

Figure 15 shows the Yamahata measured chevron motion for voltages ranging from 0 V to 1.5 V in 0.1 V steps. The maximum displacement was 280 nm, which is approximately 2 pixels. Each data point is the average of 10 pictures taken at each voltage. The data was fit to a parabolic curve with a fit of  $R^2 = 0.9999$ .

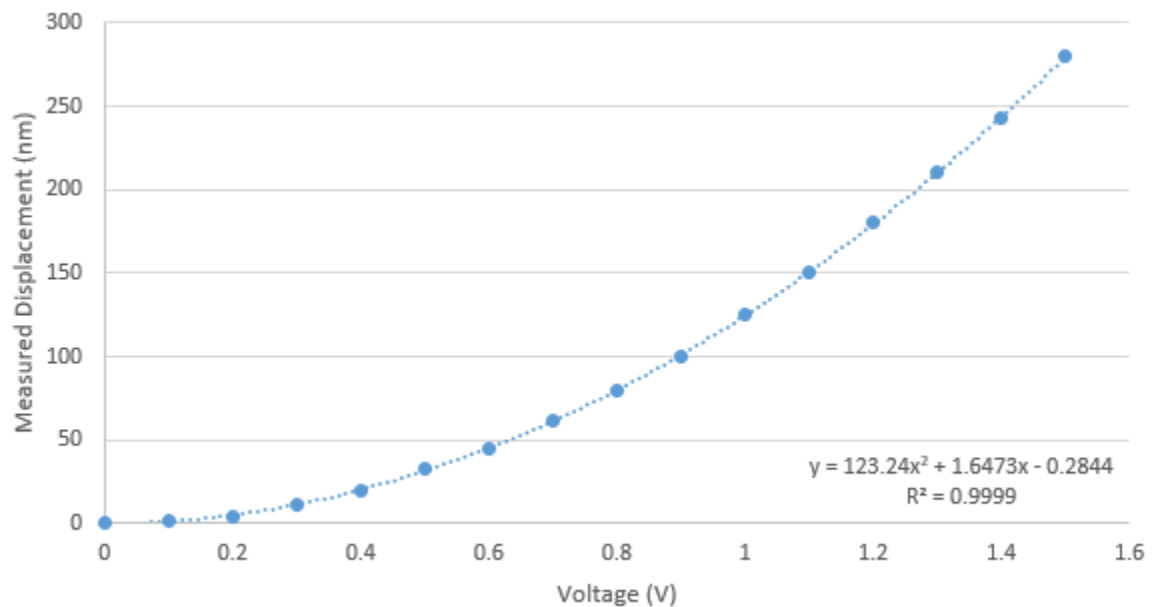


Figure 15: Yamahata results of applying a 0.1 to 1.5 V load to the chevrons.

Figure 16 shows a zoom in from 400 mV to 600 mV. Figure 16 also includes error bars for the measurements, showing the range of one standard deviation of the 10 measurements. In this case the average standard deviation of the measurements is  $\pm 3$  nm. The error bars are not included in Figure 15 as they would be too small at the scale of this plot to be notable. For the smaller voltage range of Figure 16, the system is approximately linear (slope  $\approx 250$  nm/V) which is expected of a parabolic system, as the derivative will be linear. This will be explored further in Section 6.2.

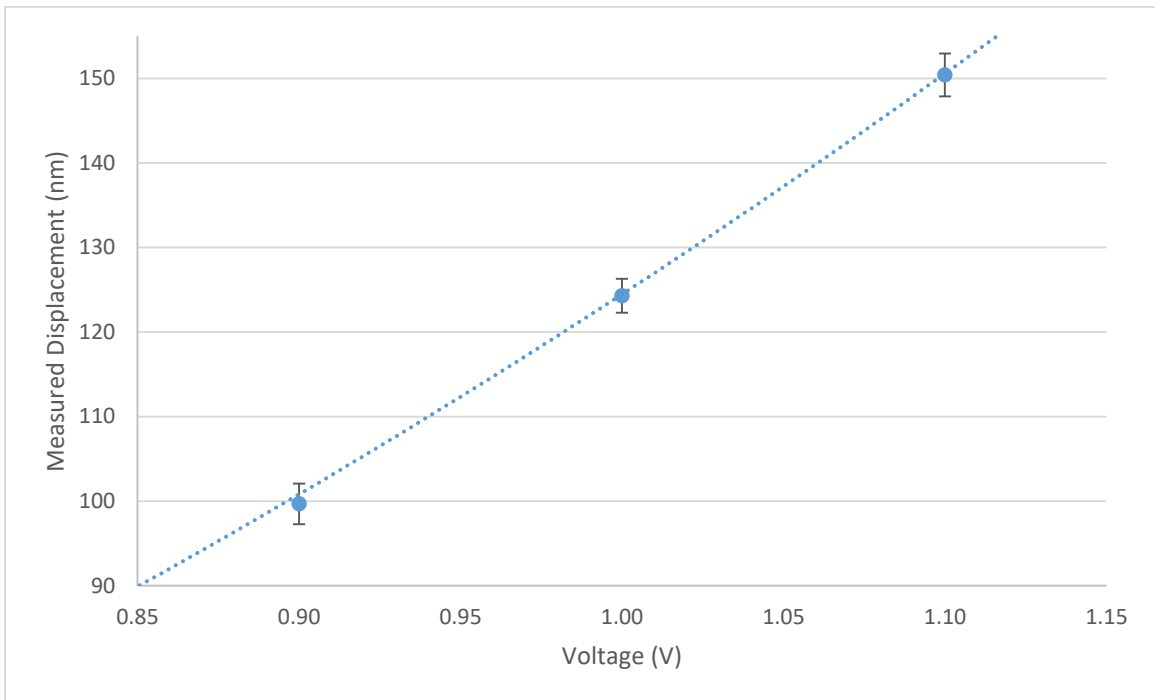


Figure 16: Yamahata results of applying a 0.9 to 1.1 V load to the chevrons.

#### 4.4. Accuracy, Precision, and Resolution

When referring to the results of tests, accuracy, precision, and resolution will frequently be referred to. Accuracy refers to how close the measured result is to the expected result, precision/resolution refers to the standard deviation of the measurement in nm or as a fraction of a pixel.

Figure 17 illustrates sample measurements, comparing accuracy and precision of measurements. The blue dots represent each of the individual measurements. The green line shows the expected result, 0 in this case. The red line indicates the average value of the measured data.

If the red line is close to the green line, the result is said to be accurate. In the top results the average value of the samples have nearly zero error compared to the expected result. In the bottom results the average value of the samples have large error compared to the expected result.

If the blue dots are all relatively close to one another, the standard deviation of the measurement will be relatively small, and the result is said to be precise. In the left results the standard deviation of the samples is small. In the right results the standard deviation of the samples is large. Thus, the top right image of Figure 17 is accurate, but imprecise. The bottom left image of Figure 17 is precise but inaccurate. Only the top left of Figure 17 is both accurate and precise.

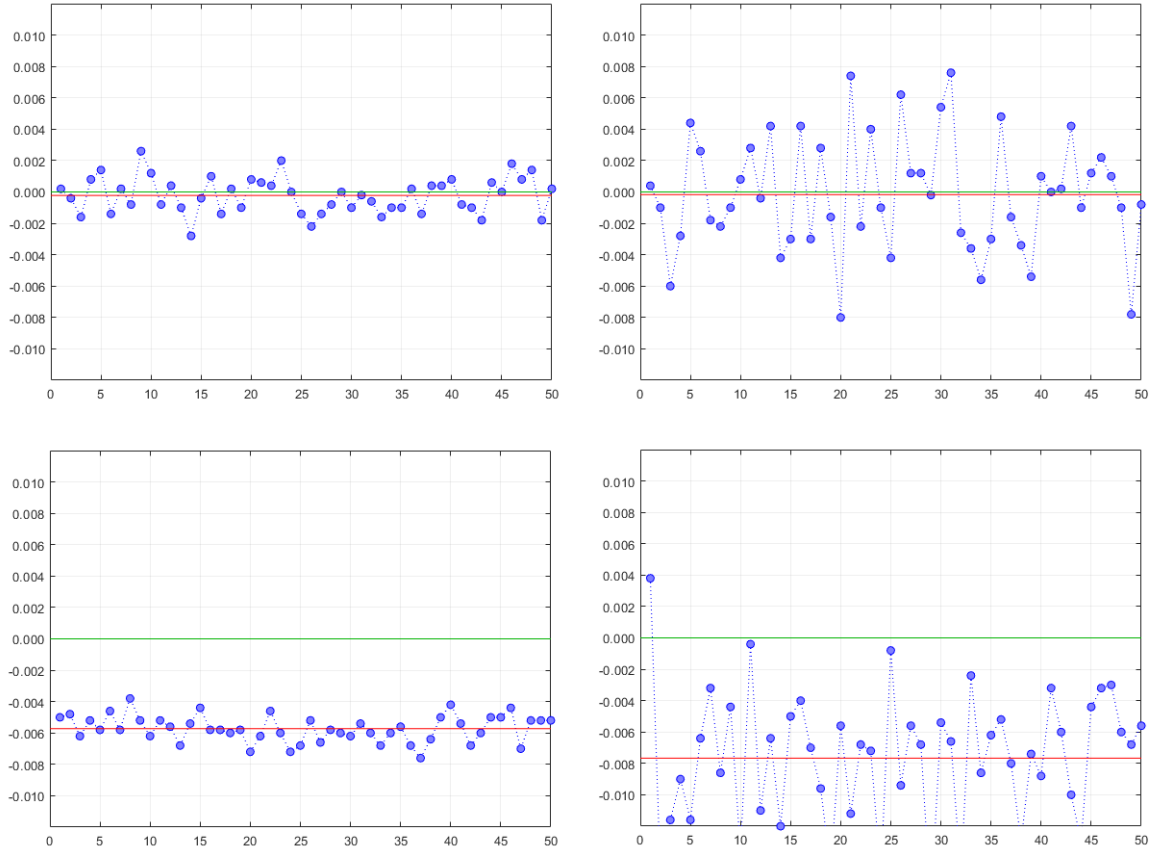


Figure 17: Precise and accurate (top left), imprecise and accurate (top right), inaccurate and precise (bottom left), and inaccurate and imprecise (bottom right) results.



## 5. Cross Correlation

This thesis develops a new measurement system that uses cross correlation in order to evaluate displacements. This chapter will cover the theory behind cross correlation, as well as mathematical sources of error within the cross correlation algorithm.

### 5.1. Theory

Cross correlation is a technique used to determine the phase shift of a signal. The signal must be converted to discrete points (in the case of the optical measurements being studied, the pixels are these discrete points). The example in Figure 18 below shows the same function twice (a segment of a sinusoid); one shifted relative to the other. Note that both functions have trailing and leading zeros. The only difference between these two functions is a phase shift of +7 points (taken for the red function relative to the blue function).

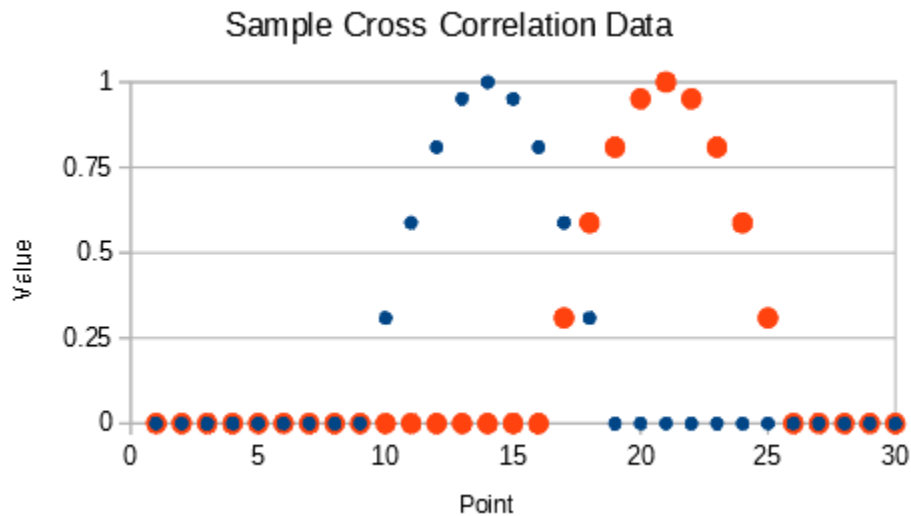


Figure 18: Two functions, one a phase shifted version of the other.

In the above example, we will let the blue function be  $f(i)$  and let the red function be  $g(i)$ . We are also treating  $i$  as a series of discrete points from 0 to  $n$  (where  $n$

= 30 in this example). In this case, the cross correlation of  $(f \star g)$  as a function of shift  $\delta$  is defined as:

$$(f \star g)[\delta] = \sum_{i=0}^n f(i) * g(i + \delta) \quad (3)$$

It is also important to understand how the end point shift works in practice. When the red curve is shifted to the right by 1 unit, for all values except 1 & 30, the new value of  $g(i)$  is simply the old value at  $g(i-1)$ . The rightmost value previously at point 30 is removed from the system. On the leftmost side, the empty value at point 1 is filled with a zero.

In order to determine how well correlated the two data sets are, each pair of corresponding points are multiplied together, and a sum is taken of the products. One of the two data samples is then shifted one point to either the left or right (both directions are used) and the sum of products is calculated again at this shift. In Figure 18 only two points have non-zero values for both signals, only points 17 & 18 overlap and the sum of their products is 0.36 in this example. As the red function shift to the right this sum will decrease, as its shift to the left this sum will begin to increase, then eventually decreasing as the red function shifts beyond the blue function.

Once sums for all shifts have been calculated, the sums are plotted. The maximum value of this plot is the phase shift between the signals. In Figure 19, the maximum value, 5, occurs when shifting 7 points to the left, indicating that shifting the red function 7 points to the left would cause the two functions to be in their closest alignment.

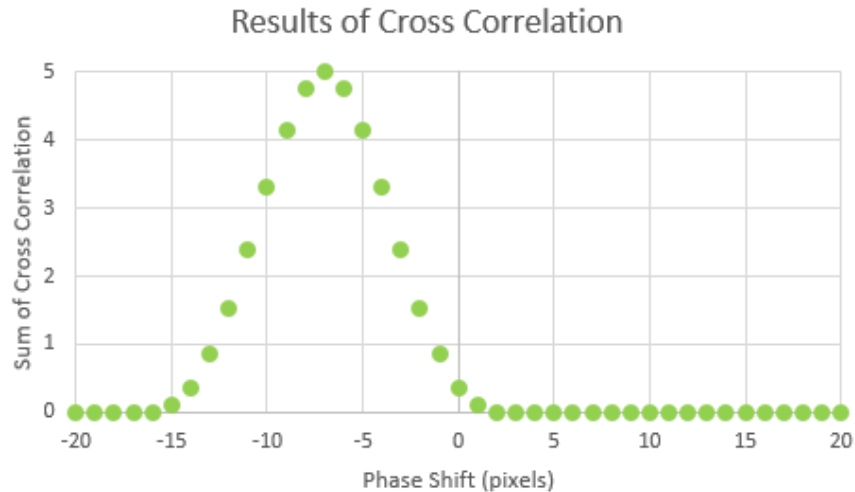


Figure 19: The sum of products for each phase shift  $\delta$  applied to Figure 18.

However, in practice image data will not be as simple as shown in the example above. There are two primary problems that will be present in practical experiments, non-discrete shifts and noise. Non-discrete shifts are a result of a signal that does not shift to an exact data point. In the case of the images being taken, the pixels are 163 nm wide. If the sample moves 424 nm, this will be a shift of 2.6 pixels. However, the peak value of the cross correlation will occur at 3 pixels, as the cross correlation function is only able to evaluate discrete shifts. Noise also presents problems. The random nature of noise will cause variations in the peak of the correlation even when the data is supposed to be constant.

## 5.2. SSSIG & MIG

Not all images are equally suited to DIC. A flat image will convey no information; the more contrast, the 'sharper' the image, the more accurate the DIC process will be. Two common metrics of image contrast are SSSIG and MIG. SSSIG refers to the Sum of Square of Subset Intensity Gradients (see Pan 2010 [31]). For a subset of  $N \times N$  pixels with intensity =  $g(i,j)$ , the SSSIG in the  $x$  direction is given by:

$$SSSIG_x = \sum_{i=1}^N \sum_{j=1}^N [g_x(i,j)]^2 \quad (4)$$

where  $g_x$  is the local gradient calculated by central differences. Similarly for  $SSSIG_y$ . The total  $SSSIG$  is the vector sum of  $SSSIG_x$  and  $SSSIG_y$ .

For a 1D case for a subset of  $N$  pixels with intensity =  $g(i)$ , the  $SSSIG$  is given by:

$$SSSIG_{1D} = \sum_{i=1}^N [g_x(i)]^2 \quad (5)$$

MIG refers to the Mean Intensity Gradient (see Pan 2010 [31]). For an image  $H$  x  $W$  pixels with intensity =  $g(i,j)$ , the MIG is given by:

$$MIG = \frac{1}{(H \times W)} \sum_{i=1}^N \sum_{j=1}^N |\nabla g(i,j)| \quad (6)$$

Where the local magnitude of the gradient vector is  $|\nabla g| = \sqrt{g_x(i,j)^2 + g_y(i,j)^2}$ .

In the case of x motion:

$$MIG_x = \frac{1}{(H \times W)} \sum_{i=1}^N \sum_{j=1}^N |g_x(i,j)| \quad (7)$$

For a 1D case for a subset of  $N$  pixels with intensity =  $g(i)$ , the MIG is given by:

$$MIG_{1D} = \frac{1}{N} \sum_{i=1}^N |g_x(i)| \quad (8)$$

SSSIG is a local metric, MIG is whole image. If an  $N \times N$  image is approximately uniform, then:

$$SSSIG_x \cong N^2 MIG_x^2 \quad (9)$$

For a 1D case for a subset of  $N$  pixels:

$$SSSIG_{1D} \cong N MIG_{1D}^2 \quad (10)$$

### 5.3. DIC errors

There are two main types of error in DIC. The first is called systemic or bias error and is caused by the sub-pixel interpolation (see Tu [11] and Bing [32]). Bias error manifests as a roughly sinusoidal error that is 0 at 0, 0.5, 1.0 pixels and extremum at approximately 0.25 and 0.75 pixels. The interpolation is exact at integer pixels and is symmetric about the mid pixel values but is an approximation at intermediary values. The mean value of the bias error is zero. The magnitude of the bias error  $\sigma_{bias}$  is inversely proportional to the square root of the SSSIG:

$$\sigma_{bias} \propto \frac{1}{\sqrt{SSSIG}} \quad (11)$$

The units of bias error are pixels and can be reduced to millipixel levels or lower depending on SSSIG. Averaging multiple images does not reduce the bias noise. Filtering the data, for example via a Gaussian or moving mean average does not reduce the bias error (see Mazzoleni [33]). Bias error can be reduced by using higher order interpolation schemes or by using iterative methods such as Newton-Raphson (Wang 2016 [34]).

The second type of error is called deviation error and is caused by random variations from image acquisition, for example camera noise. Let the standard deviation of noise in the image be given by  $\sigma_n$ . The standard deviation error  $\sigma_{dev}$  is given by:

$$\sigma_{dev} \propto \frac{\sqrt{2} \sigma_n}{\sqrt{SSSIG}} \quad (12)$$

The  $\sqrt{2}$  occurs as  $\sigma_n$  is present in both the reference and new image.

The noise  $\sigma_n$  is expressed in the image bit depth. The units of deviation error  $\sigma_{dev}$  are pixels and can be reduced to millipixel levels or lower depending on  $\sigma_n$  and SSSIG.

Since  $\sigma_n$  is random, averaging  $N$  images reduces the bias noise by a factor of  $\sqrt{N}$ . Filtering the data, for example via a Gaussian or moving mean average may or may not reduce the bias error. There is a trade off as smoothing the data will reduce the noise level but will also reduce the SSSIG. Filtering with a short span filter can be beneficial if the noise is higher spatial frequency while the SSSIG content is longer spatial wavelength (see Pan 2013 [35]).

## 6. Cross Correlation Simulation

To progress from the previous, simple example, a MATLAB simulation was constructed which would use a more realistic signal, as well as allowing for a variety of imperfections to be introduced in order to see what impact these factors would have on the results of the cross correlation. These simulations will also help to develop techniques for reducing the effects of these imperfections. Figure 20 provides a sample of the type of periodic structure used, and different aspects of it will illustrate various elements which can cause measurement errors in experiments.

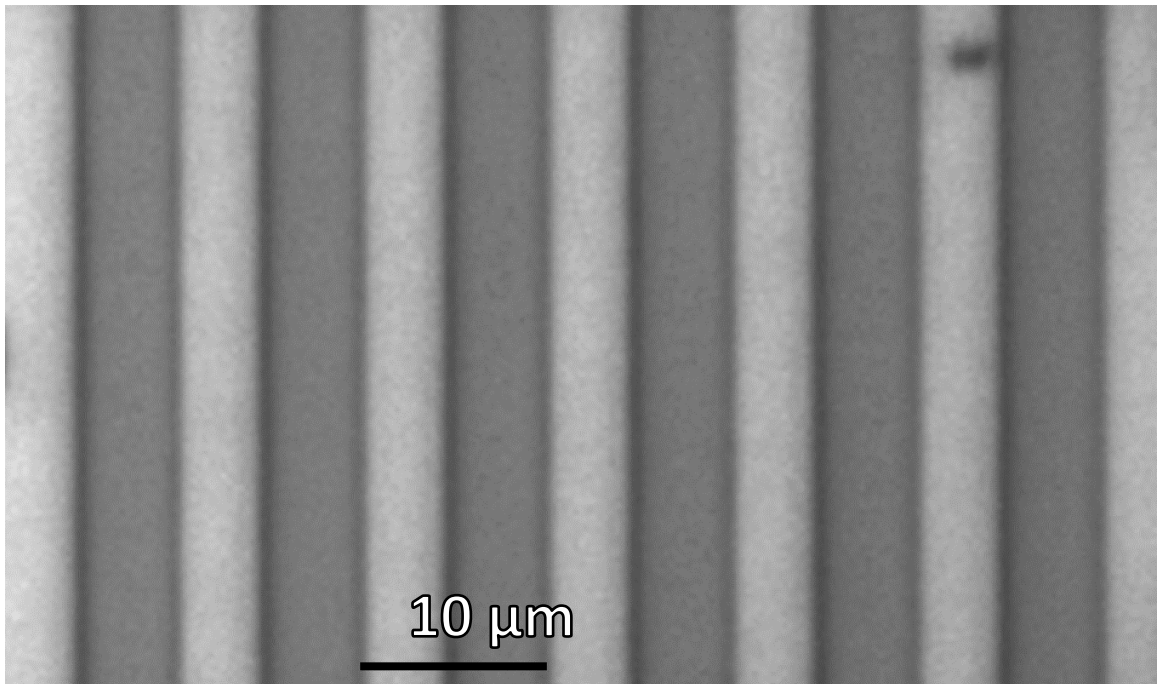


Figure 20: An ROI from a sample chip

### 6.1. Measuring Sub-Pixel

Figure 21 shows a truncated sinusoid approximation of the 1D profile of the periodic structure in Figure 20. Note that while colour values in the 8-bit image vary from 0 to 255, the simulated profiles will range from 0 to 1. Additionally, the colour values in 8-bit images must be discrete, integer values, the simulated values can vary continuously from 0 to 1. In this case, the flat regions at  $y = 0$  are the darker regions, and

the flat regions at  $y = 1$  are the lighter regions. The slopes indicate the transitions between these regions.

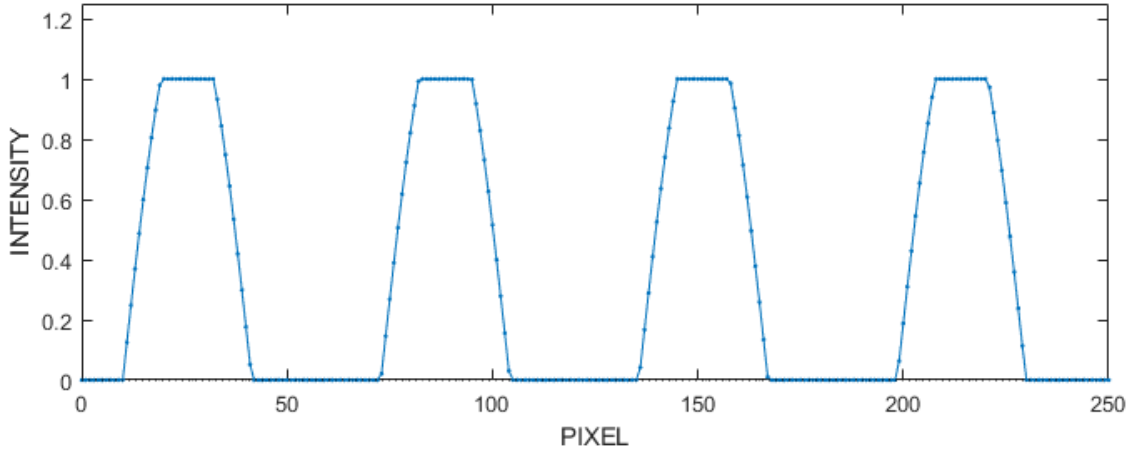


Figure 21: Truncated sinusoid approximation of Figure 11, used for simulation purposes.

By creating a second identical function and phase shifting it a known amount, tests can be conducted on the cross correlation algorithm. Figure 22 shows a sample which has been shifted (in red) compared to the original function from Figure 21 (in blue). This shift can be varied by the user. The phase shift in Figure 22 is 4 pixels.

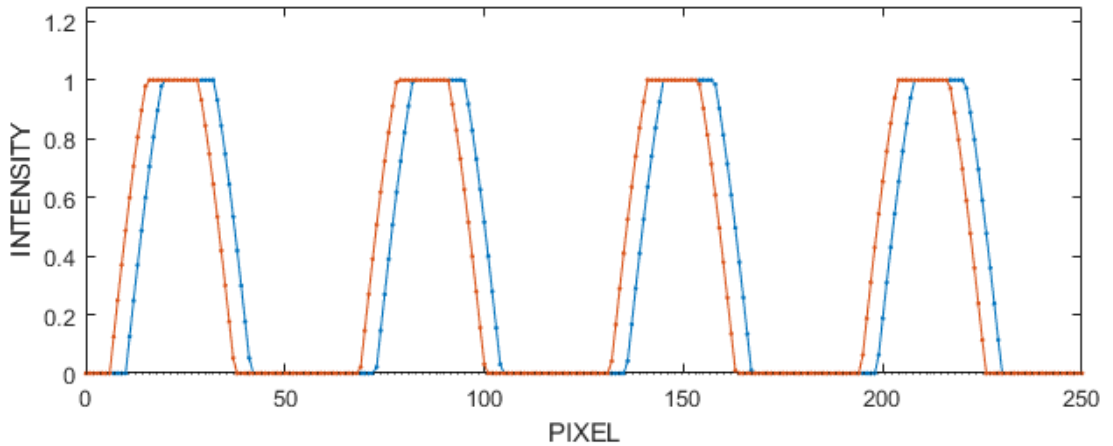


Figure 22: Same simulated function as in Figure 21, with a phase shifted equivalent in red. The shift in this example is four pixels.



Figure 23 indicates the cross correlation results of this test. A peak value is found at a shift of 4 pixels, and for every spatial wavelength from that point there are subsequent lower peaks. The outer peaks decrease in height as the ends are padded with zeros. The second half of the image shows a zoom in of the peak value.

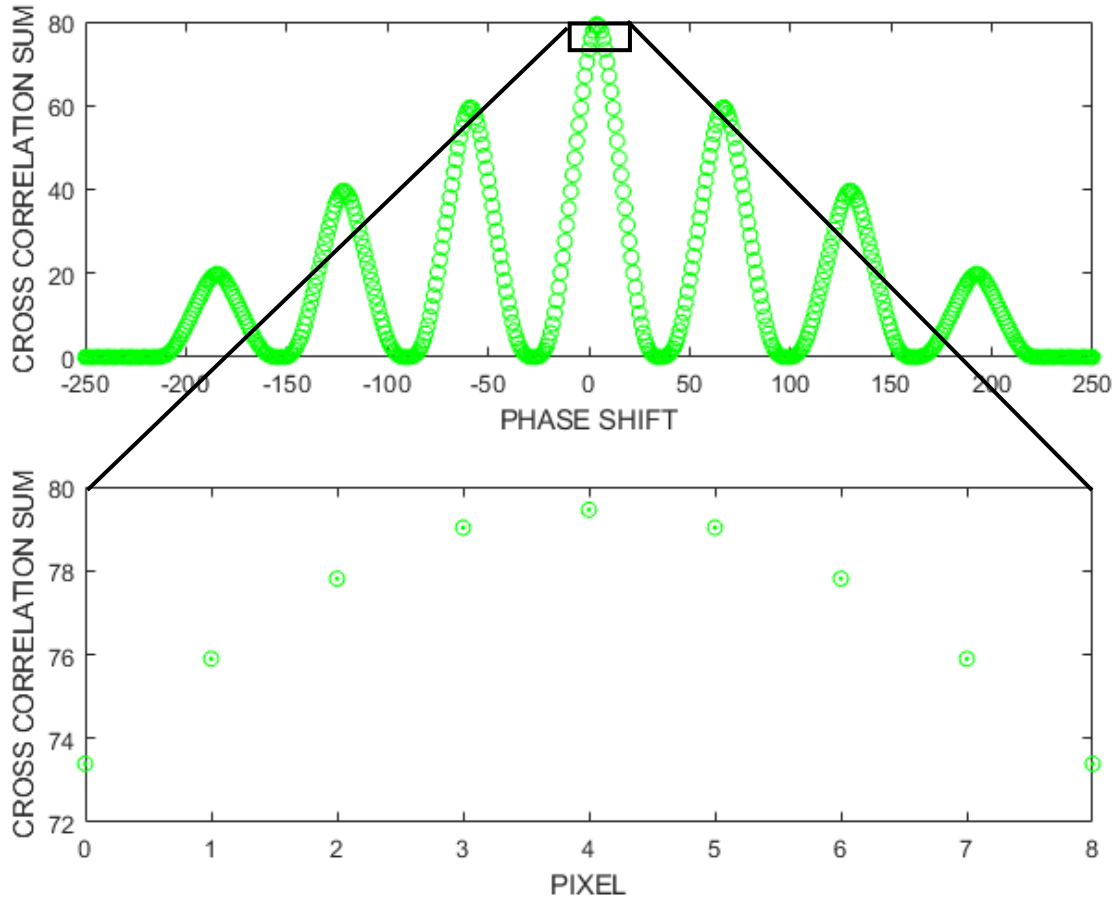


Figure 23: The results of the cross correlation (above) and the region of the peak value (below).

The previous cross correlation example was only able to measure shifts of discrete steps, or to the nearest pixel in the case of images. In order to accomplish sub pixel correlation, the cross correlation results curve is first calculated on the original profile with pixel accuracy. The peak value is determined, identifying the phase shift to the nearest pixel. Figure 24 shows the cross correlation results using a phase shift of 4.3 pixels. A peak value is found at a shift of  $\sim 4$  pixels, but not exactly at 4 pixels. In order to obtain measurements with greater precision than a single pixel, the resulting cross correlation is upsampled. Upsampling describes techniques which will create

additional data points from a discrete set of data. This algorithm will add a user defined number of points between real points. The pixel level cross correlation values were stored in a vector. A subset vector near the peak of the correlation curve (peak  $\pm 4$  pixels) is extracted. This subset is expanded and intermediate sub pixels values are interpolated using a cubic spline fit. The peak of the interpolated vector is the measured displacement of the system. Figure 24 bottom shows the 0.1 pixel interpolated cross correlation results: a peak value is found at a shift of 4.3 pixels, as this is the point at which the maximum value of the cubic spline is found. The user can select the sub pixel resolution. Figure 25 shows sample plots of the spline fits of 1, 0.1, and 0.01 pixels.

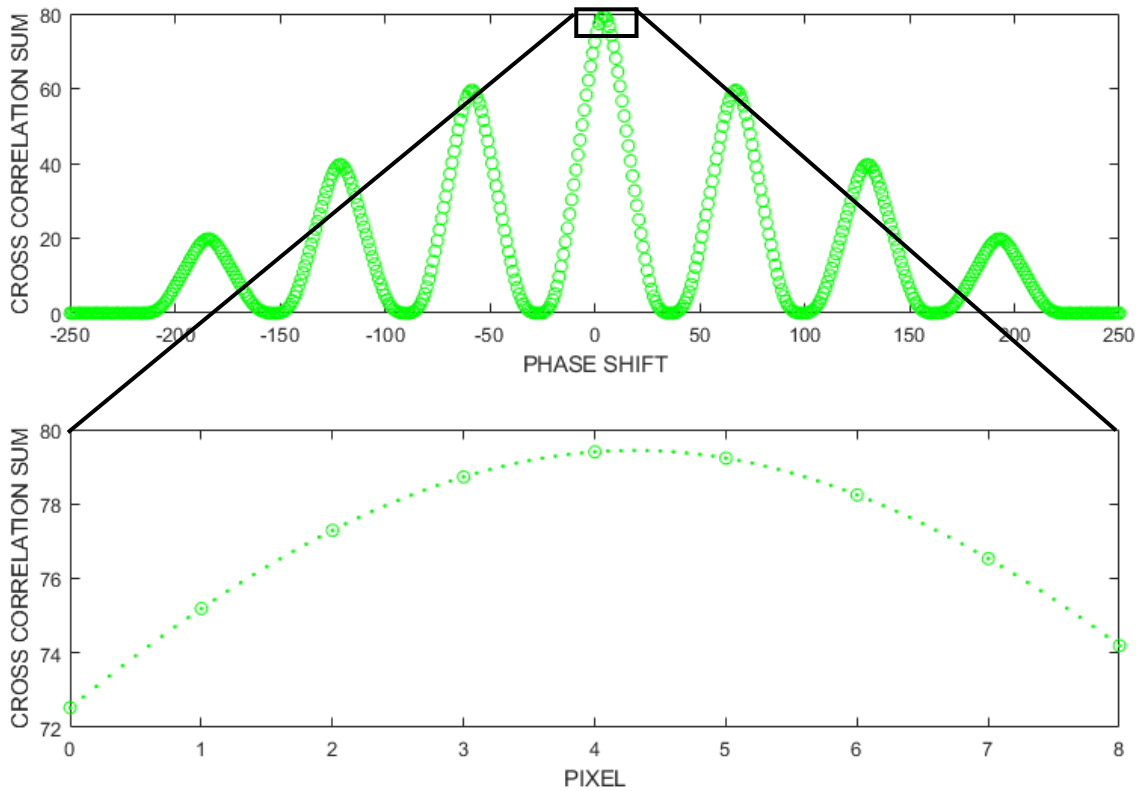


Figure 24: The results of the cross correlation (above) and the region of the peak value (~4.3 pixels), with a 0.1 subpixel spline fit (below).

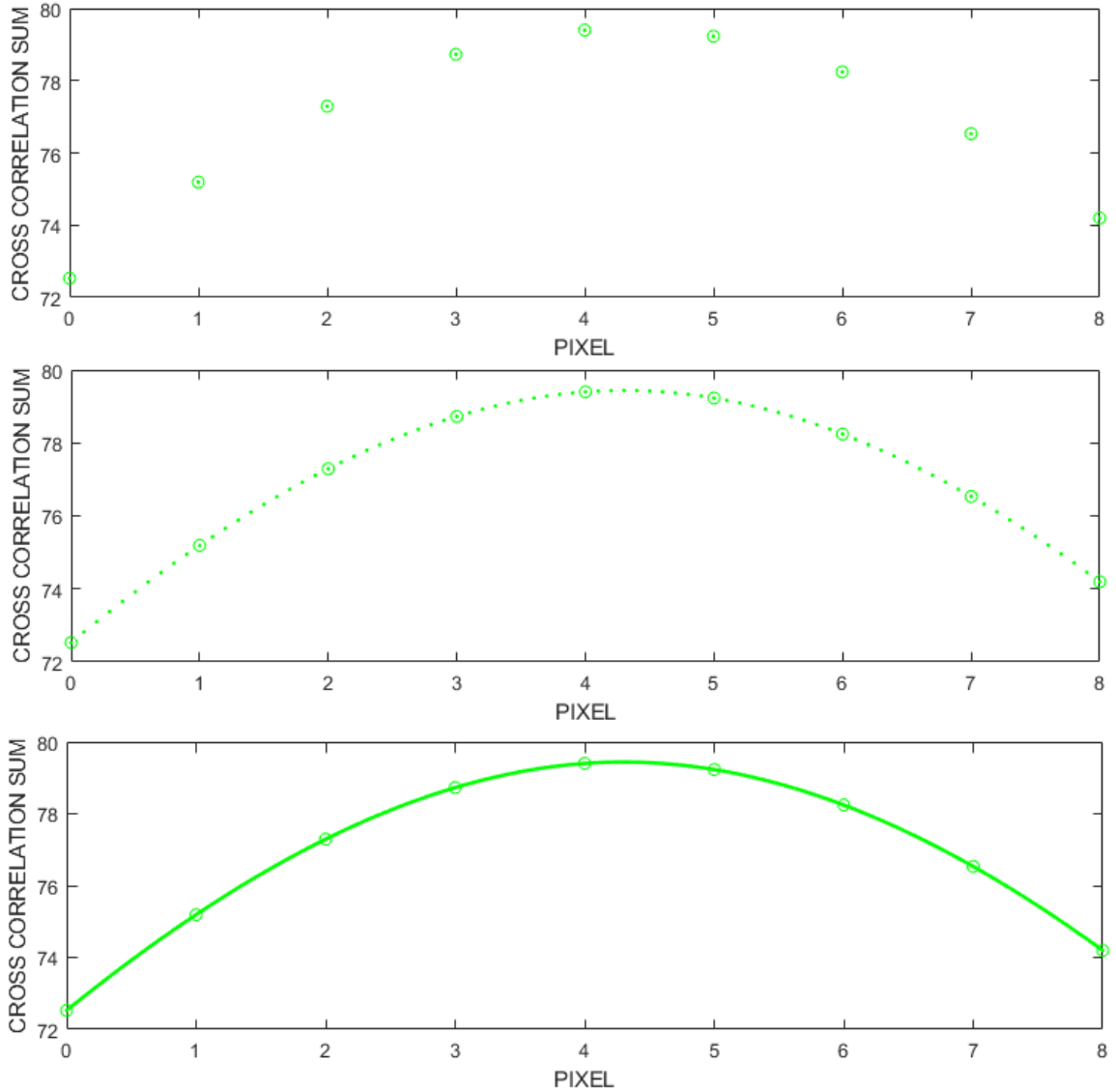


Figure 25: Plots showing different levels of sub-pixel upsampling, showing 1, 1/10, and 1/100 pixel samples.

The results of sub pixel interpolation were tested for various input shifts. For the simulation tests, the user defined shift of the second function was arbitrarily set to 4 pixels initially, then 4.5, 4.7, 4.73, 4.732, and finally 4.7623. The measured displacement of the simulation at each offset tested with each spline are recorded in Table 2.

Table 3 shows the difference between the actual displacement and the measured displacement. Values in green have errors of  $\sim 0$  as these measurements occur exactly on the pixel or the half pixel. Values in red have errors larger than the sub pixel resolution.

From Table 2, the sub pixel interpolation from 0.5 pixels down to 0.01 sub pixels calculates the correct shift to within the sub pixel resolution. From 0.001 pixels onward however, this is no longer true. The error becomes greater than the resolution (0.002-0.003 pixels), and subsequent increases in the resolution of the measurement do not provide improved results.

Applying this result for the entire spectrum of phase shifts between 4 and 5 pixels, a curve can be found which represents the systemic or bias error in the measurement resulting from the spline fit. Figure 26 shows the deviation between the input and measured displacement as the displacement was changed from 4 to 5 pixels. The blue points were discretized to 1/5000 pixels. This increased resolution is used because a discretization to 1/1000 pixels would produce stepped results that are multiples of 0.001. This would produce a series of jumps, as opposed to the smoother curve seen in Figure 26.

When the displacement range was changed to, for example, from 2 to 3 pixels or 9 to 10 pixels, the same bias curve was observed. For the function used in the simulations, the peak bias error was on the order of 3 millipixels, however when measuring small displacements, the bias error will be much smaller. For example when measuring nm level displacements (e.g. from 4 to 4.02  $\mu\text{m}$ ), the motion will be confined to the initial part of Figure 17 and the bias error is less than 0.1 millipixels. The bias error is a result of the spline upsample of the cross correlation being an approximation of the real values.

Table 2: Simulated displacement measurements at varying resolutions.

	<b>Sub-Pixel Resolution</b>					
	1	.5	.1	.01	.001	.0001
<b>Displacement</b>	<b>Measured Displacement</b>					
4	4	4.0	4.0	4.00	4.000	4.0000
4.5	4	4.5	4.5	4.50	4.500	4.4995
4.7	5	4.5	4.7	4.70	4.702	4.7023
4.76	5	5.0	4.8	4.76	4.763	4.7629
4.762	5	5.0	4.8	4.76	4.765	4.7649
4.7623	5	5.0	4.8	4.77	4.765	4.7652

Table 3: Error in measurement between expected and measured displacement at various resolutions.

	<b>Sub-Pixel Resolution</b>					
	1	0.5	0.1	0.01	0.001	0.0001
<b>Displacement</b>	<b>Measured Displacement ERROR</b>					
4	0.000	0.000	0.000	0.000	0.000	0.000
4.5	-0.500	0.000	0.000	0.000	0.000	0.000
4.7	0.300	-0.200	0.000	0.000	0.002	0.002
4.76	0.240	0.240	0.040	0.000	0.003	0.003
4.762	0.238	0.238	0.038	-0.002	0.003	0.003
4.7623	0.238	0.238	0.038	0.008	0.003	0.003

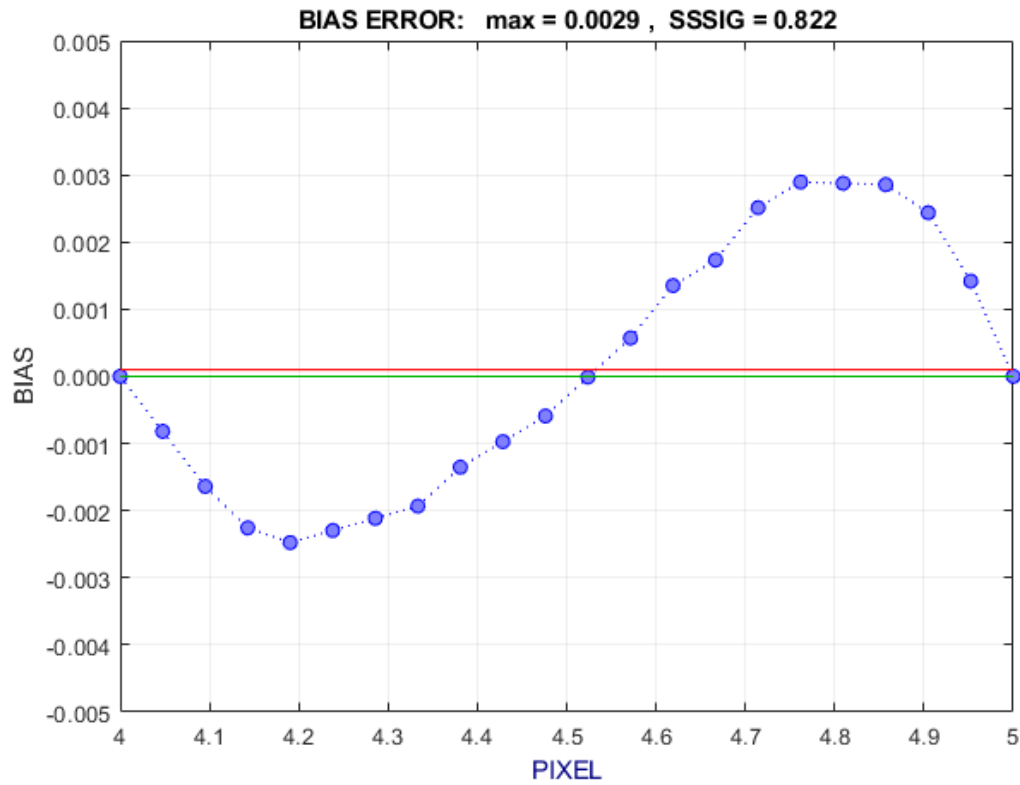


Figure 26: The deviation of the spline fit as the actual displacement ranges from 4 to 5 pixels. Y axis units are pixels.

## 6.2. Noise

Noise is the uncontrollable variation in the data collected from a given sample which is created by the imprecision of the measuring device. In the case of the images, a given pixel in different images will not maintain a consistent colour value with respect to time, even if the image does not move. As a sample, Figure 27 shows a zoom in of a small section of a MEMS device. There is a small variation in the colour between nearby pixels on the same body, and while much of these differences will be eliminated by column averaging, some degree of noise will remain.

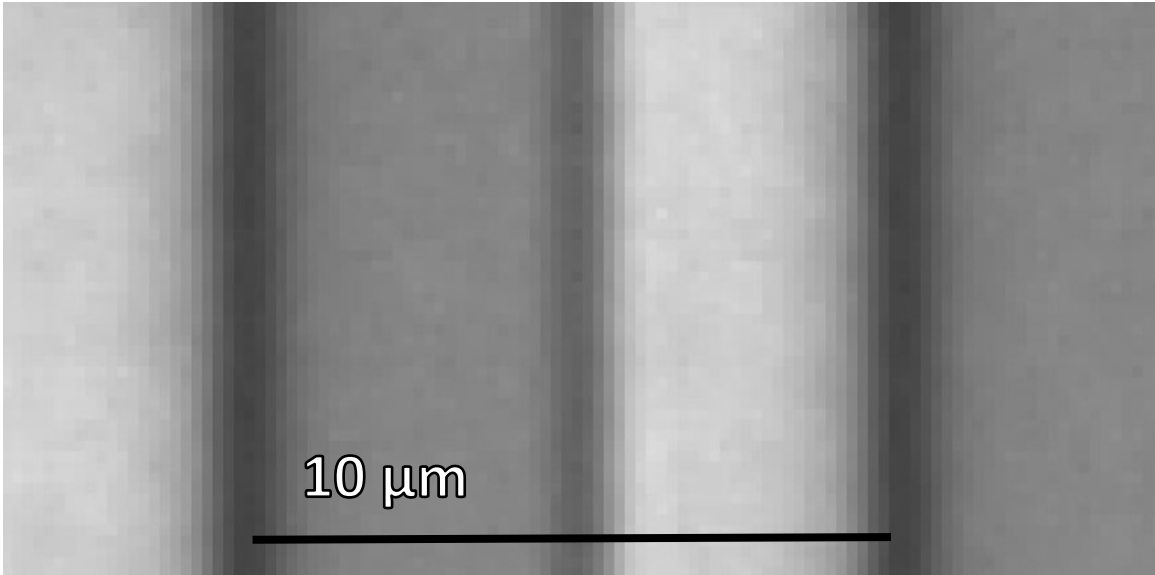


Figure 27: A zoom in on Figure 20, showing the noise in the image.

In order to simulate this effect, small values of noise ranging from  $+X$  to  $-X$ , where  $X$  is a user defined value, are added to the existing data to generate two profiles as shown in Figure 28. The noise follows a uniform distribution. Running the same simulation will yield slightly different results each time, so multiple tests were conducted and averaged. Simulations were then run with different levels of noise in order to determine its impact. Table 4 shows the result of simulations in which noise values of 0.01 to 0.15 were used. Simulations were conducted using an input phase shift of 4 pixels, which should result in a measurement of 4 pixels, as discussed in the previous section. The spline fit of these tests is discretized to 0.001 sub-pixel resolution.

The results indicate that the standard deviation will, on average, improve with more measurements, although individual test results will vary. The results in Table 4 show the average standard deviation from 100 tests using a given number of measurements. Results using more than 50 measurements did not tend to produce notably better results.

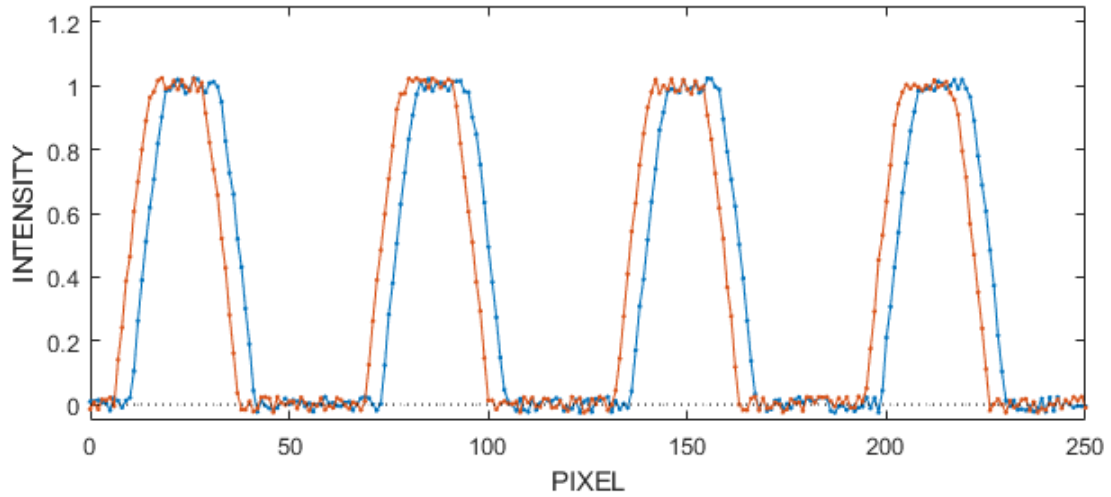


Figure 28: A sample of noisy data based on Figure 27, with a maximum noise of 0.02 units.

Table 4: Expected deviation of measurements of simulated data with noise.

Noise Value	$\sigma$ (4 measurements)	$\sigma$ (10 measurements)	$\sigma$ (50 measurements)
0.00	0	0	0
0.01	0.00015	0.00011	0.0005
0.02	0.003	0.002	0.001
0.03	0.005	0.003	0.001
0.05	0.009	0.005	0.003
0.10	0.019	0.011	0.005
0.15	0.032	0.020	0.009

Figure 29 and Figure 30 detail the results from two sets of 50 measurements. In both images, the green line represents zero error, and the red line is the average value of all 50 data points. Figure 29 has a noise value of 0.003, and Figure 30 has a noise value of 0.010. For a noise value of 0.003, the measurements had a standard deviation of 0.001 pixels, and for a noise value of 0.010 the measurements had a standard deviation of 0.005 pixels.



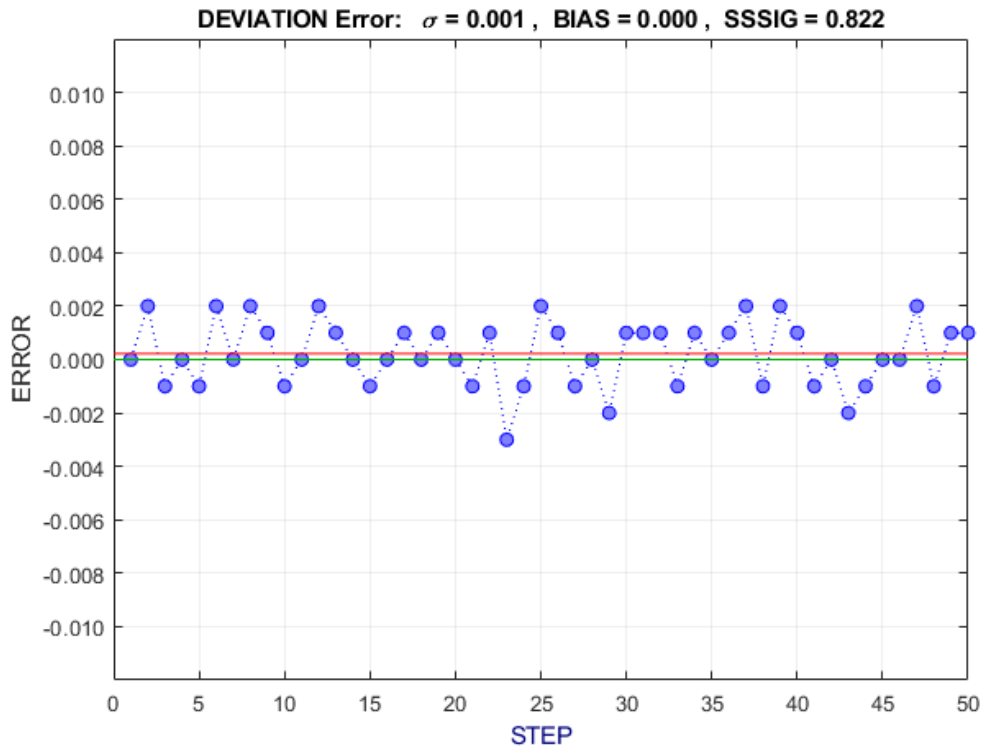


Figure 29: Pixel deviation error for simulated images with a noise value of 0.003. Y axis units are pixels.

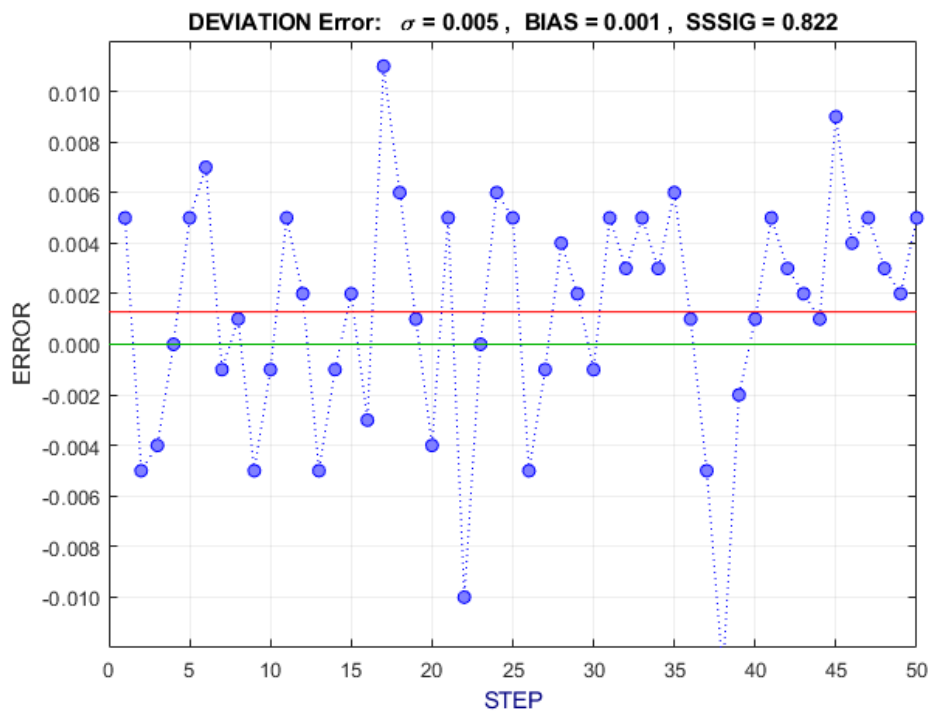


Figure 30: Pixel deviation error for simulated images with a noise value of 0.010. Y axis units are pixels.

Figure 31 plots the measured standard deviation vs. noise from 0.001 to 0.010 for two different truncated sinusoidal signals: one with a SSSIG of 0.7 and a steeper signal with a SSSIG of 1.35 (sine truncated lower). In both case the standard deviation increased linearly with noise and the higher SSSIG image had lower noise (~70% of the noise for a ~doubling of SSSIG).

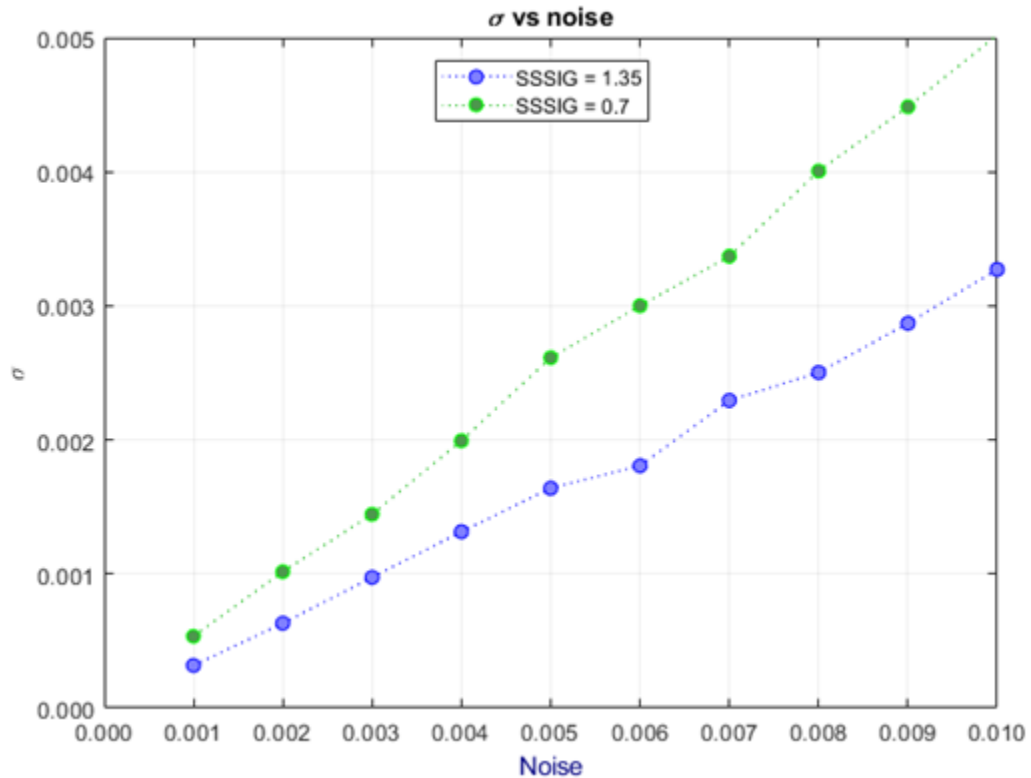


Figure 31: Pixel deviation error from noise values of 0.001 to 0.010.

### 6.3. Offset

The offset of a data set is the displacement along the vertical axis relative to 0. In the images being analyzed, 0 is treated as pure black. In general, the images will not reach this level of darkness. Likewise, the lighter regions do not typically reach a pure white. For example on an 8-bit image (0-255) used in these experiments, the data will

typically range from a low value of 50- 100 and a high value of 200 - 220. Vertical offset will create problems for the cross correlation function. As a function is shifted to the left, the leftmost value is removed from the function, and a zero is added on the right end. If the points on the left and right extremes of the function are zero, no change in the product sum occurs, but if these values are non-zero, then there may be spurious changes in the cross correlation.

Figure 32 shows a simulated offset, with all values being increased by a vertical offset of 0.1 (where the simulated signal varies from 0 - 1). This test does not include the noise from the previous tests. By applying this offset to the original data set, the results in Table 5 were computed. These results indicate that an increase in the vertical offset corresponds to an increase in the bias error. This is due to the trailing zeroes used in the cross correlation algorithm. As such, Table 5 indicates that it is essential to remove any vertical offset. To account for this in the algorithm, a user-defined number of points at the beginning of the dataset are averaged and subtracted from all points to remove the offset. The number of points being averaged can be adjusted by the user.

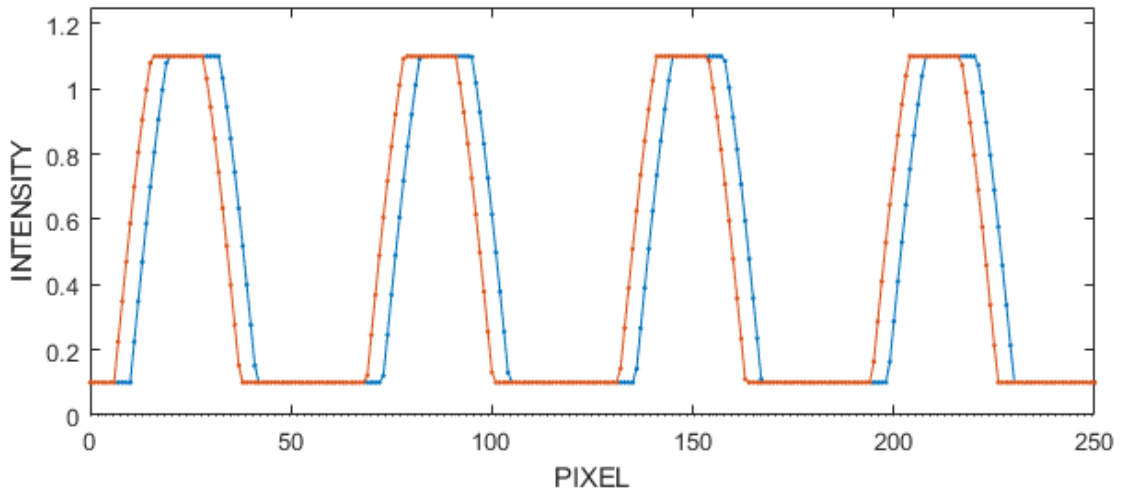


Figure 32: An offset simulation with an offset value of 0.1.

Table 5: Measured displacements of simulated data with vertical offset.

Offset Value	Result	Bias Error
0.00	4.000	0
0.05	3.996	0.004
0.10	3.982	0.018
0.15	3.960	0.040
0.20	3.929	0.071
0.25	3.889	0.111
0.30	3.840	0.160

An additional point of interest is that the magnitude of the error does not appear to vary depending on the number of pixels phase shifted. A displacement of 2, 4 or 10 pixels will see virtually the same error, provided the same disturbance is applied to the system. If an offset of 0.1 is shown to cause a certain change in the result when the curve is shifted by 2 pixels, this will also be the case when it is shifted by 10 pixels.

#### 6.4. Slope

In the processed images, an uneven light source can sometimes create cases where one side of the image experiences a greater intensity of light than the other. Adjusting the illumination in the setup can help to reduce this but eliminating it completely has proven to be difficult. Figure 33 shows an exaggerated example of this on a real image. The left side of the image is much lighter than the right side, and this will impact the results of the measurements. Converting this to a simulated dataset, Figure 34 presents an approximate model of how slope effects the data.

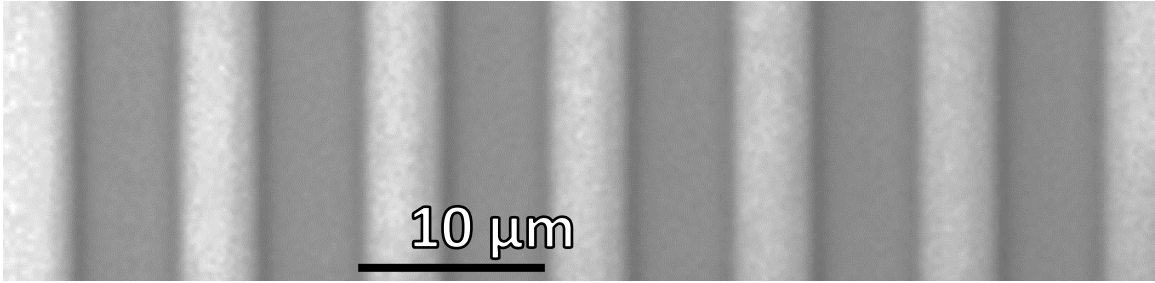


Figure 33: A higher contrast, and cropped version of Figure 20, showing how the left side of the image is lighter than the right.

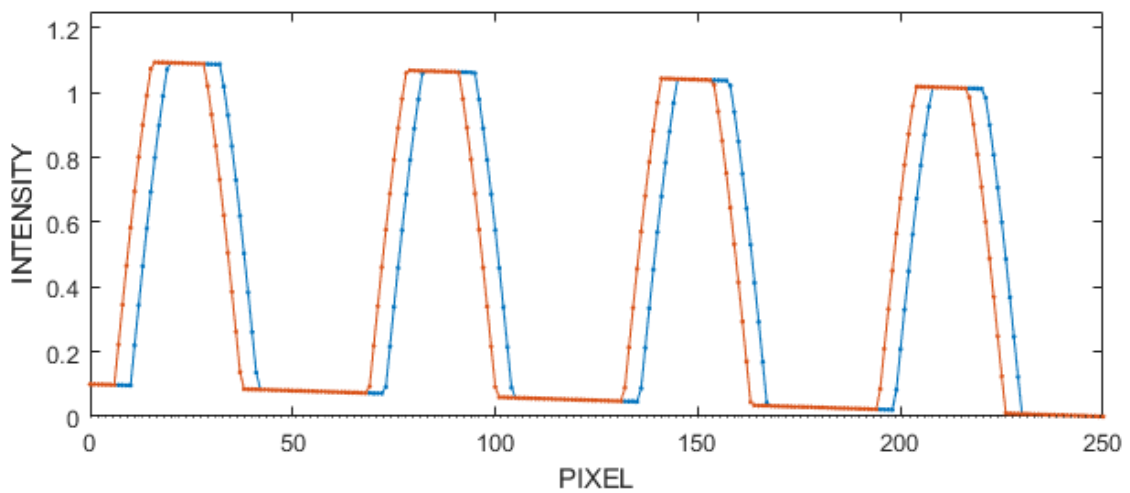


Figure 34: A simulation of the slope result, based on Figure 33 with a slope of 0.10.

The “slope value” is the change in intensity from the beginning of the function to the end. In Figure 34, this corresponds to -0.1 as the initial dark region starts at a value of 0.1 and finishes at a value of 0. Table 6 shows the results of this test: the errors of the changing slope value are approximately half that of the offset for the same value. Positive and negative slopes produce the same magnitude of errors. Thus, it is important to remove any slope by ensuring even illumination and mathematically removing any remaining slope. For the purposes of the developed software, the first few points at the beginning and end of the dataset are averaged and a slope trendline calculated. This slope is then proportionally subtracted from all points.

Table 6: Measured displacements of simulated data with a slope.

Slope Value	Measurement	Pixel Error
0.00	4.000	0
0.05	3.998	0.002
0.10	3.991	0.009
0.15	3.980	0.020
0.20	3.965	0.035
0.25	3.945	0.055
0.30	3.920	0.080

### 6.5. Uneven Illumination

It is also possible that the illumination can result in two different slopes occurring within a result. It was often found that correcting the slope of a real image would result in either the light or the dark regions being corrected, with the other retaining a reduced but non-zero, slope. Figure 35 shows an exaggerated example of this on a real image. Converting this to a simulated dataset, Figure 36 presents an approximate model of how an uneven illumination effects the data. The results from this test are presented in Table 7: uneven illumination does not affect the measurements.

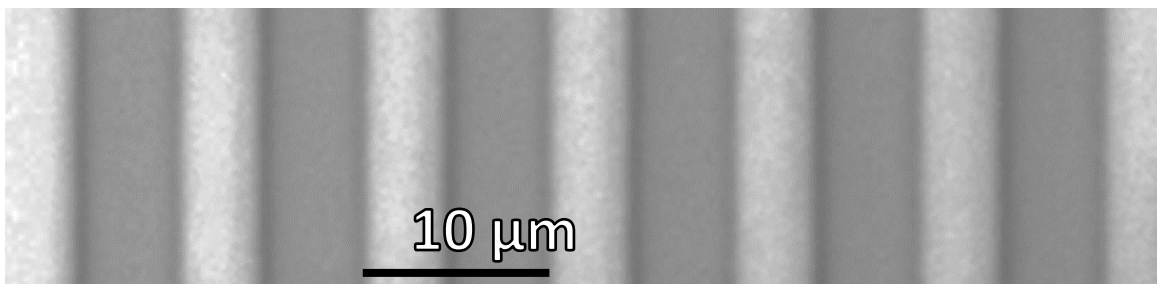


Figure 35: Figure 33 again. This time note that the brightness of the light regions varies more than the dark regions.

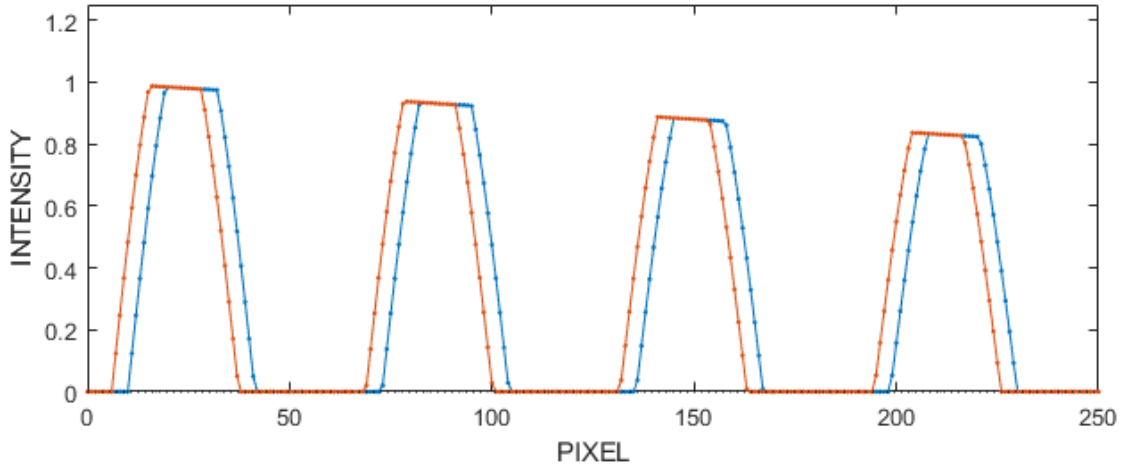


Figure 36: A simulated representation of an uneven illumination.

Table 7: Measured displacements of simulated data with an uneven slope.

Uneven Slope Value	Measurement	Measurement Error
1.00	4.000	0
1.25	4.000	0.000
1.50	4.000	0.000
1.75	4.000	0.000
2.00	4.000	0.000
3.00	3.999	0.001

## 6.6. Summary

The simulations indicate that the cross correlation method of measuring displacements can measure sub-pixel motions. The level of sub-pixel discretization can be set by the user. Further discretizing beyond 0.001 pixels does not appear to provide a more accurate result. In addition, image noise will degrade the measurements in a linear manner. The magnitude of the error does not appear to vary depending on the number of pixels phase shifted.

There are a number of factors to consider when implementing the algorithm. Most importantly, the data is required to start and finish at a value of zero and any offset should be removed from the dataset. Secondly any slope should be removed from the dataset.



## 7. Experimental Setup

### 7.1. Microscope

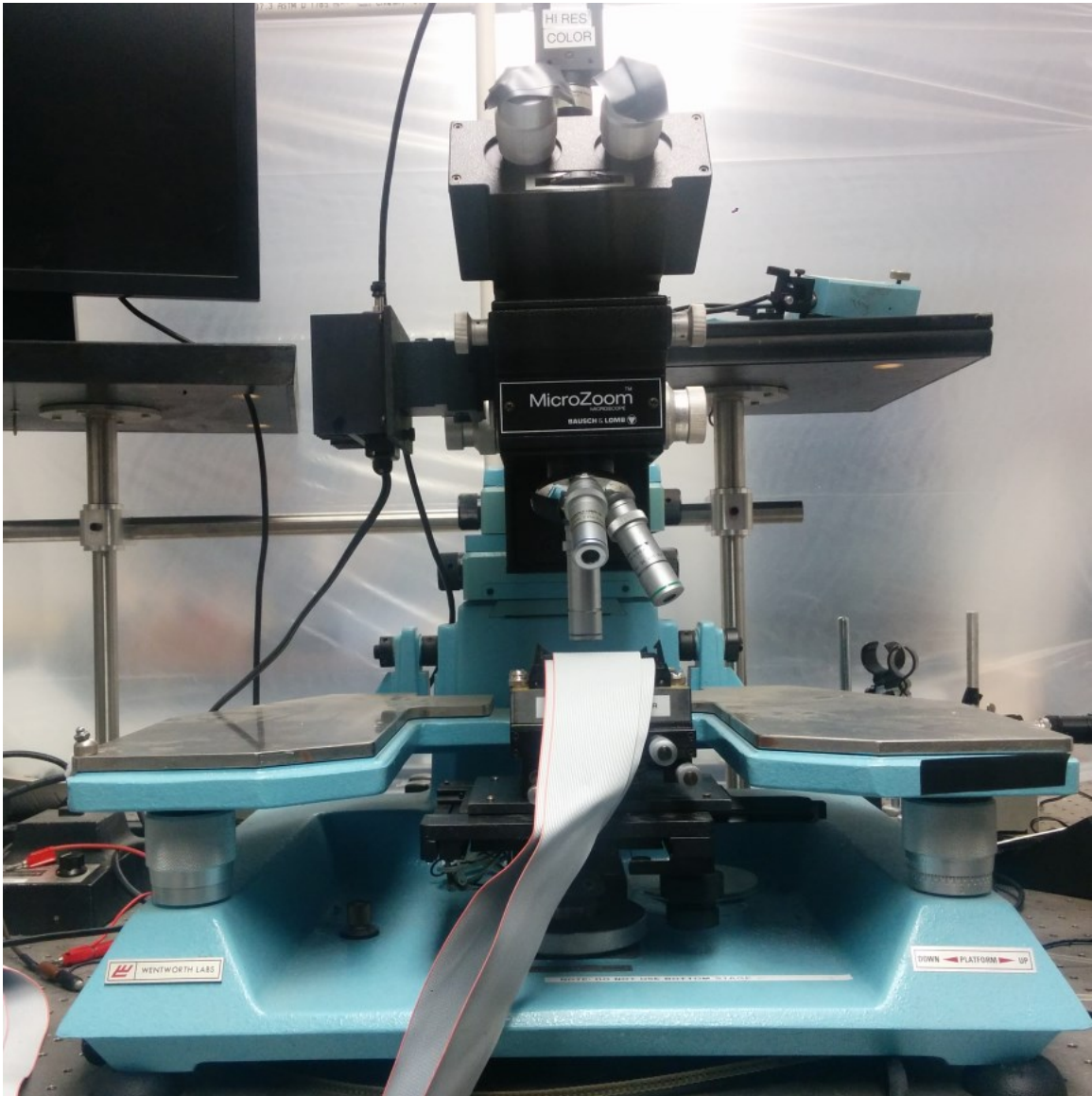


Figure 37: Microscope and camera setup

The microscope, seen in Figure 37, used was a Wentworth Probestation M901 which included a Bausch & Lomb MicroZoom Microscope. The microscope has three microscope objectives which provide 2.25x, 8x, and 25x magnification, though the 2.25x lens is rarely used. In addition, a zoom knob on the main body of the microscope allows for variations from 1x to 2x in addition to what is provided by the main lens. This is

always set to either 1 or 2, as the analog nature of the dial makes it impossible to consistently return to any other positions.

This leads to 50x and 16x magnifications being the two most commonly used. 50x is used when zooming in on ROIs for the highest resolution images, and 16x is typically used when looking at a larger portion of a single device, which is required when the moving and reference ROI aren't immediately next to one another. 8x is also used on occasion to obtain images of entire devices on a chip.

The chip is fastened into a Zero Insertion Force socket which receives signals from 68 wires connected via 2 ribbon cables. These ribbon cables, and the chip under the microscope, are shown in Figure 38. Usually only 2-5 of these wires are used in a single experiment. The chip can be manually moved in the X and Y direction using the microscope XY stage.

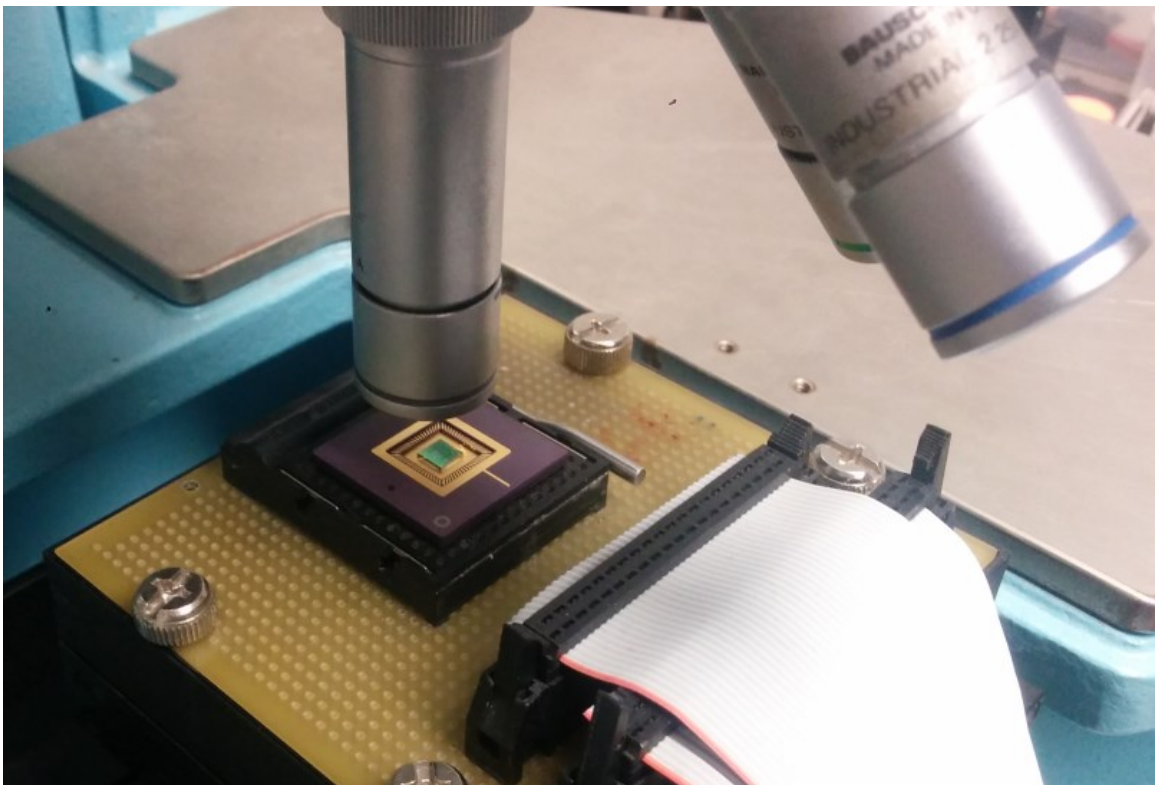


Figure 38: The chip under the microscope, with two ribbon cables connected to the chip.

## 7.2. Camera

The camera used is a Point Grey Research Firewire GRAS-14S3C, seen in Figure 39. This is an RGB camera which has a resolution of 1280x960. At a magnification of 16x, the pixel size is 509 nm, and at 50x the pixel size is 163 nm. Control of the camera is done using LabVIEW, as well as C# image capture programs that come with the camera.

The camera is attached to the microscope via a C-mount secured by a single set screw. This has presented some minor problems, as the camera has been shown to move slightly over time. A more secure set up may lead to superior results, however this was not fully investigated. There is also the potential that heating within the camera could be having an impact on the results measured [36, 37]. The heating would cause thermal expansion, which could create a relative motion between the camera and the MEMS chip. This has not been considered within the thesis.



Figure 39: The camera attached to the microscope.

### 7.3. MEMS Chip

The MEMS Chip used in the experiments detailed in this thesis are all contained on the chip (chip code IMUDTMCI) shown in Figure 40. The size of the entire device is 28.2 mm x 28.2 mm x 6.8 mm. The top face, shown, contains the 5 mm chip (green) while the bottom half contains electrical leads which are used to draw the current used to manipulate the devices on the chip.

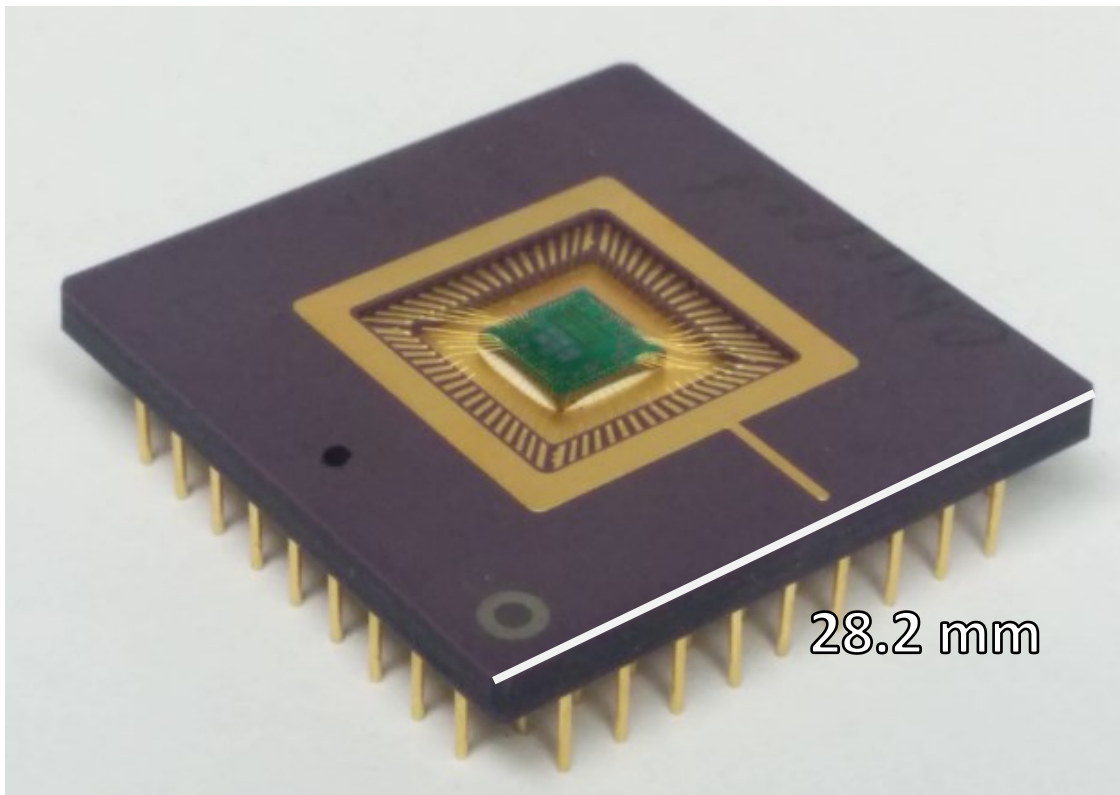


Figure 40: MEMS chip used.

Zooming in a little closer, the chip itself can be seen. This chip is 5 mm x 5 mm and contains about 30 different MEMS devices. The 68 electrical leads attach too many of these devices and are used in their actuation. Figure 40 shows a zoom in on the chip itself, with one of the MEMS devices, to be shown in Figure 41, shown in a white circle.

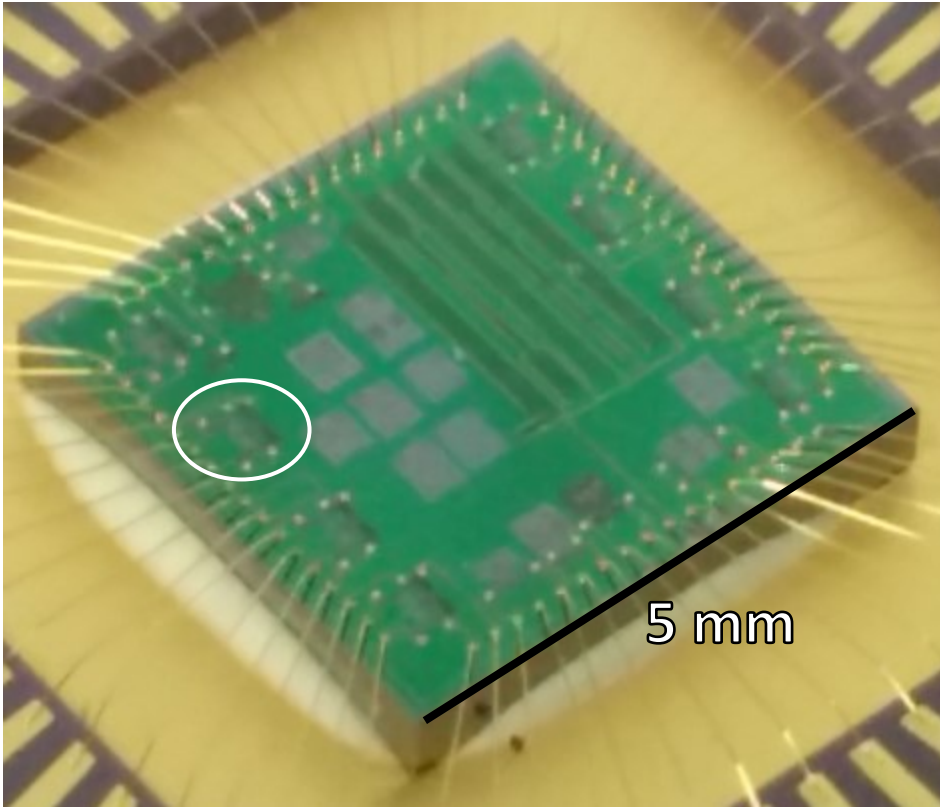


Figure 41: A zoom in showing the MEMS devices.

Figure 42 shows a microscope image of one of the primary devices used in the experiments to be discussed. The thermal actuators and varied periodic structures made it ideal for generating consistent motions and testing a variety of different ROIs. The initial function of the device was as a cell squeezer (see Barazani [38]) however it was not used in this way for the experiments in this thesis. Instead, the structures on the device were used to measure simple rectilinear displacements for the purpose of testing the newly developed cross-correlation algorithm and comparing it to the existing Yamahata algorithm in a variety of circumstances. The two stage chevron design provided a large range of motion, and the ROIs were designed to maximize their space in a 50x zoom frame. This created the most ideal case for evaluating measurements at the highest level of precision possible.

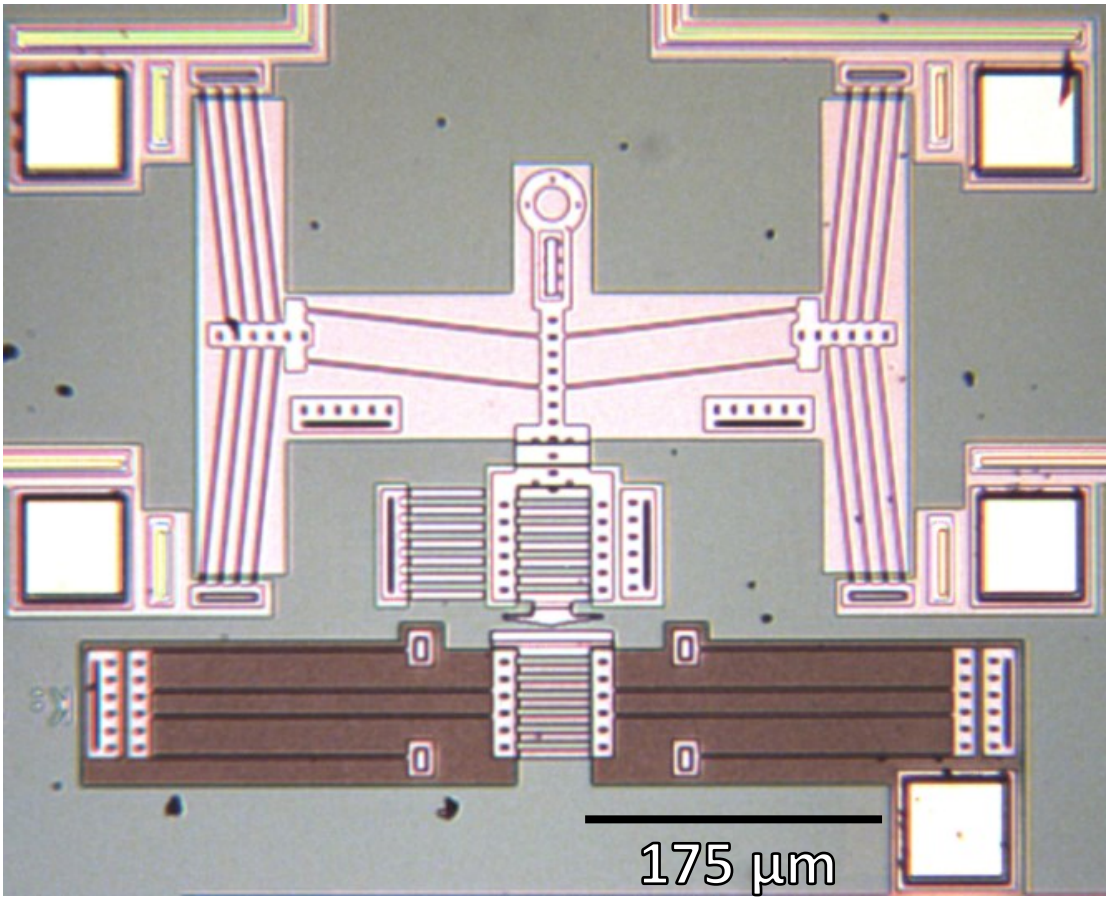


Figure 42: One of the devices on the MEMS chip.

#### 7.4. Cross-Correlation Algorithm: XCorr

A new cross-correlation algorithm, called XCorr, was written in MATLAB. The program requires minimal input from the user. It is designed for quick, efficient use. The program initially requires the user to identify a directory from which it will pull all images. After this, the user receives several prompts for program variables as indicated in Figure 43. These variables do not typically require any changes, and usually the user simply presses Enter to select the default value, but the user can replace the default value if desired:

```
Select image directory...  
  
Enter a name for the output data file (xcorr_data is default) ...  
Enter num pics per groups (10 is default) ...  
Enter sub pixel resolution (0.001 is default) ...
```

Figure 43: User prompts in the XCorr software.

Once these values have been confirmed, the user must then choose moving and reference ROIs using the mouse. The two windows in Figure 44 and Figure 45 are presented in sequence. First, in Figure 44, the moving ROI must be selected, as is indicated by the blue rectangle seen near the center of the image. This process is repeated in Figure 45 where a reference ROI is selected. In this case, it can be seen in the blue rectangle in the lower center of the image. There is also a user controlled flag in the program code to skip the choosing of ROIs and use preselected ROIs, allowing the user to enter a fixed set of ROI x, y coordinates for repeated use of the same ROIs.

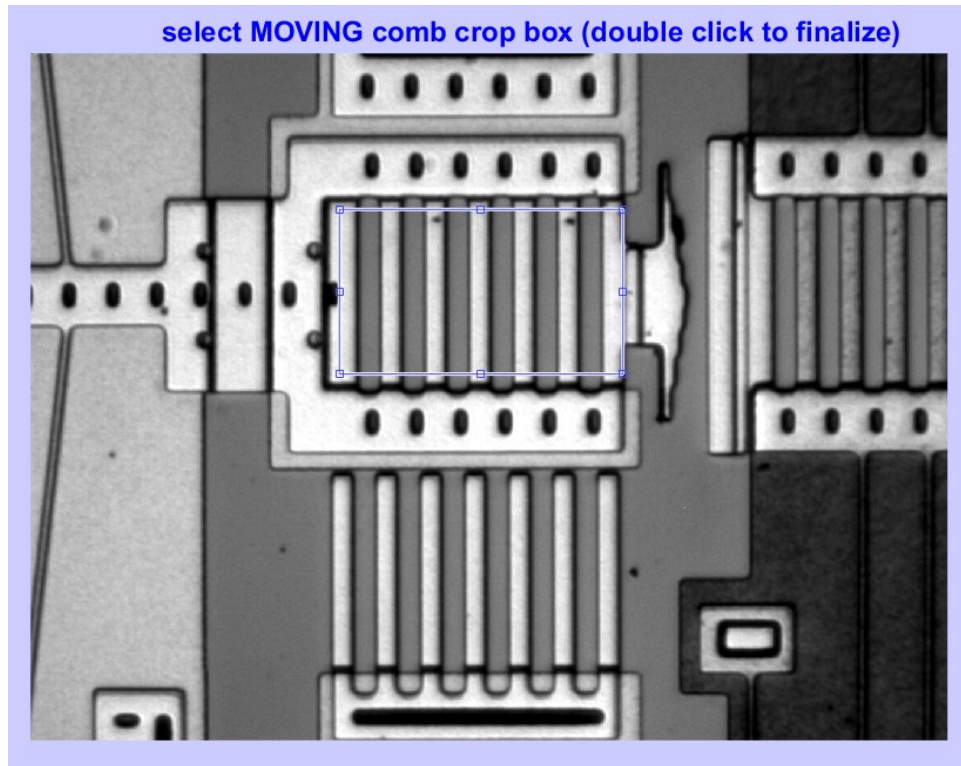


Figure 44: Choosing the moving ROI.

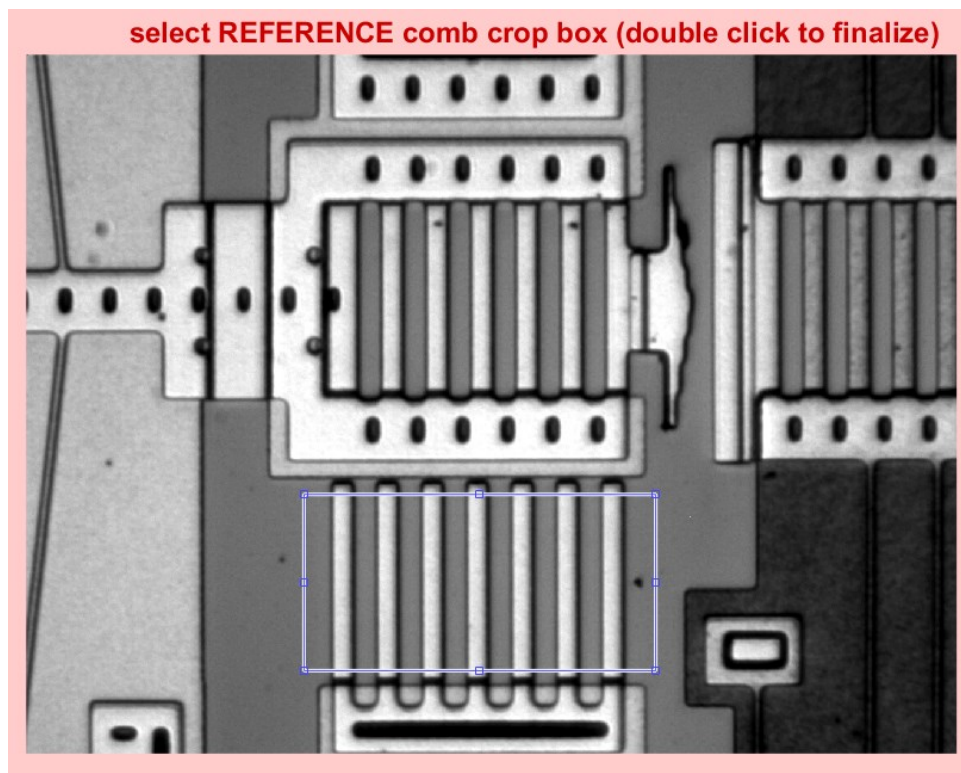


Figure 45: Choosing the reference ROI.



Once the ROIs have been selected, N images are read from the image files and all cropped to the same ROI.

- 1) For Image<sub>1</sub> to Image<sub>N</sub> each image is column averaged to produce raw profile<sub>i</sub>.
- 2) For each profile<sub>i</sub> remove profile offset
- 3) For each profile<sub>i</sub> remove profile slope

Figure 46 shows sample raw and corrected profiles for the first and last images of a 2 nm motion. Figure 47 shows a zoom in of Figure 46 demonstrating that the first and last profiles are in fact slightly different (~2 nm for 163 nm pixels). The red profile represents the first image, and the blue profile represents the final image.

The corrected profile<sub>i</sub> is then cross correlated with profile<sub>1</sub> to produce cross correlation C<sub>i</sub>. The peak of C<sub>i</sub> is found to determine the pixel level shift. The peak  $\pm 4$  points of C<sub>i</sub> are then interpolated by a cubic spline and the peak of the interpolation is the subpixel displacement. The cubic spline interpolation is done in MATLAB, as shown in Appendix A-4. Figure 48 shows the correlation produced by Figure 46 profiles. In Figure 48 a subpixel resolution of 0.1 pixels was used for clarity, in practice a subpixel resolution of 0.001 would be used.

After these calculations are completed, the results are saved to a csv file. The csv file includes information specific to the trial, including the dimensions of the ROI, the number of images, the motion of both the moving and reference ROI, the net motion, and the average motions and standard deviations of motions.

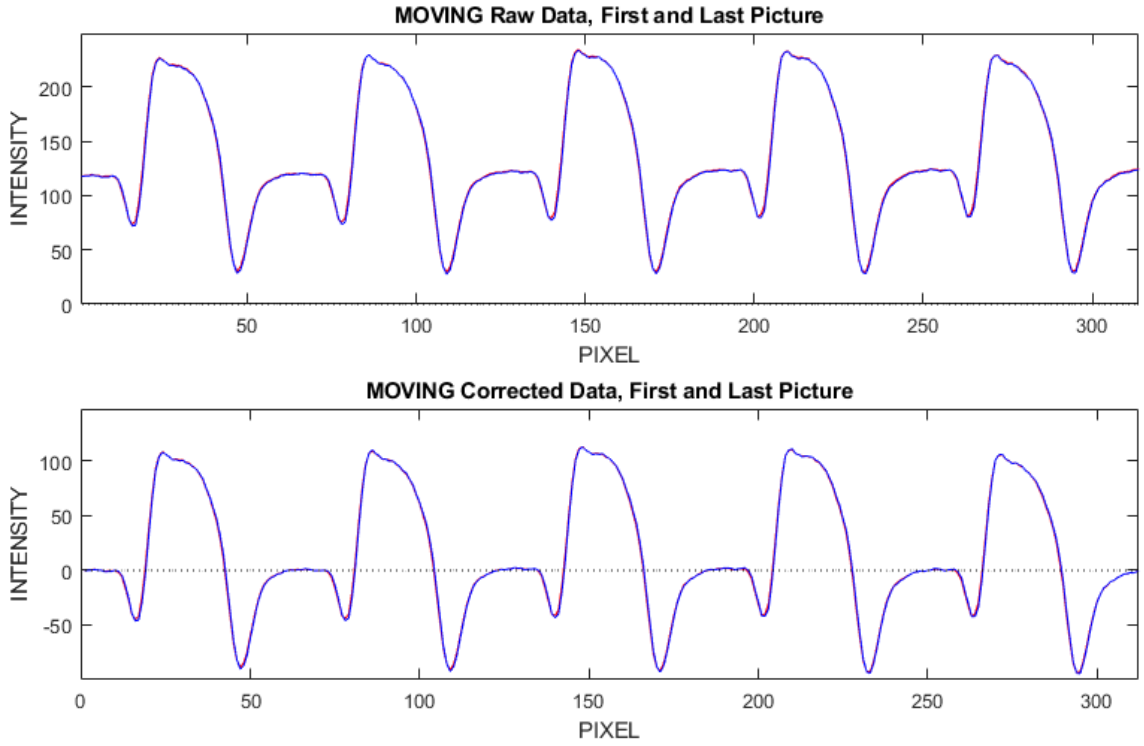


Figure 46: Raw (above) and adjusted (below) data for the first an (red) and last (blue) images

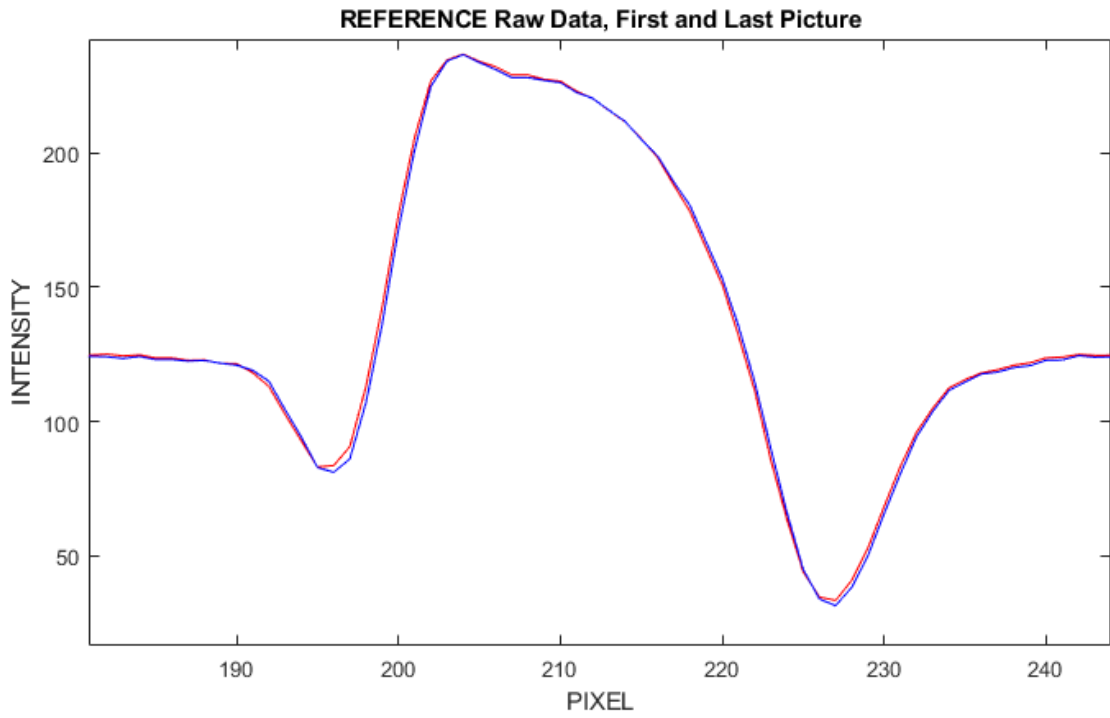


Figure 47: Zoom in on Figure 46, showing the difference between the first (red) and last (blue) images.

163 nm pixel,  $\sim 2$  nm motion.

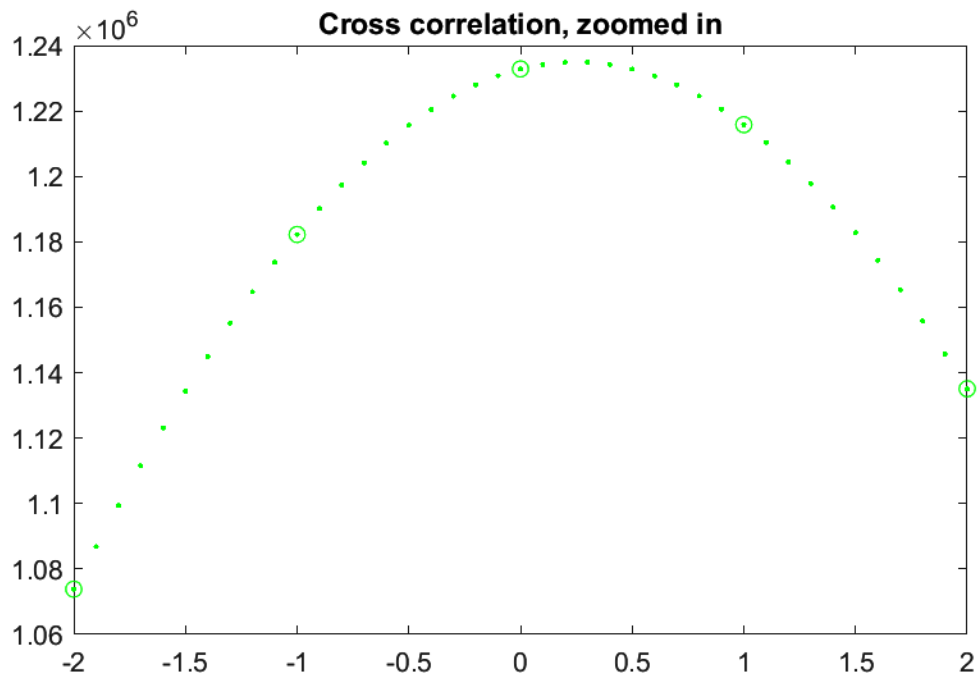
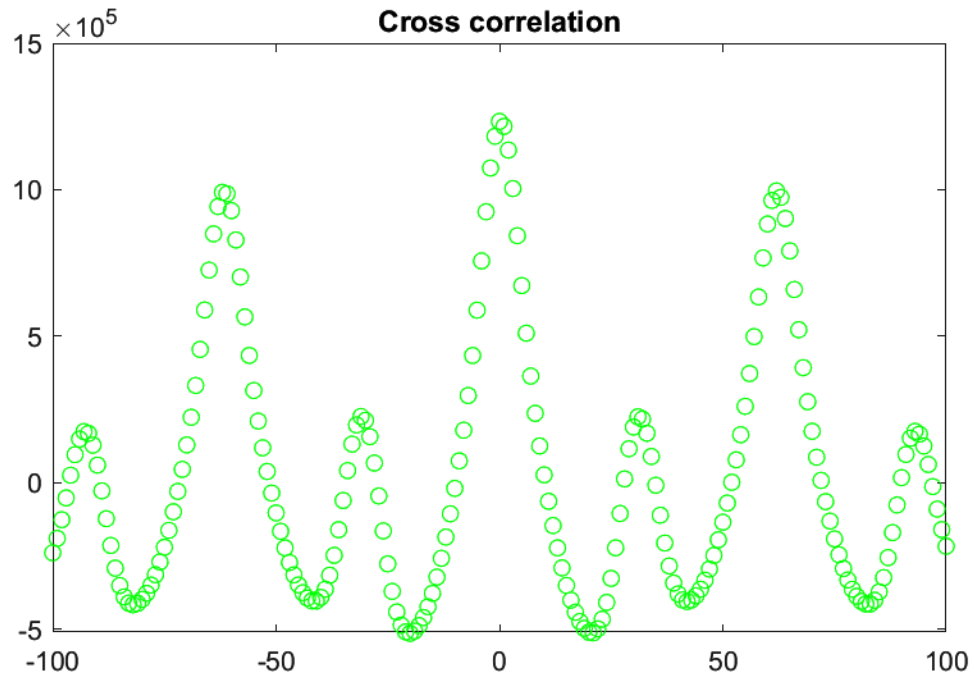


Figure 48: Cross correlation of data from Figure 43 (above) and a zoom in on the peak value and 0.1 pixel spline fit (below). Y axes are arbitrary units. X axis units are pixels

## 8. Optical Tests

In order to properly compare the two algorithms, a quantifiable metric of “goodness” is required. It is typical in these rectilinear measurement algorithms for standard deviation  $\sigma$  of a static system to be used. In some cases in the literature different multiples of standard deviation are used (often  $2\sigma$  or  $4\sigma$ ) (see Yamahata [21] and Kokorian [24]). In order to compare the XCorr and Yamahata algorithms, the standard deviations of their measurements will be compared.

### 8.1. Yamahata vs. Xcorr (Chevron)

#### 8.1.1. Test Procedure

Two primary tests were run when testing the basic functionality of the Yamahata and the XCorr algorithms. The first of these was a test of the precision of the measurements. In this, the moving and reference comb are both kept stationary, with no load applied to the moving comb. A sequence of 50 images is taken, and ROIs are selected. The same ROI is used for both the Yamahata and XCorr tests. Each image has a measurement taken relative to the original, and an array of values is created. From this, the standard deviation for each of the algorithms is calculated, and comparisons can be made.

The second test focuses on the ability of each algorithm to evaluate a step input, and what the limit is at which a step can be seen. By alternating the voltage applied, the load can move back and forth, causing shifting motion. These shifts in motion will be much less than a pixel in order to show the scale on which these algorithms can evaluate motion.

All tests for the XCorr were conducted using a  $1/1000 = 0.001$  sub-pixel resolution on the spline fit. This means that a measurement is only precise to the nearest thousandth of a pixel, or 0.163 nm. Tests were also run with  $1/2000$ ,  $1/5000$  and  $1/10,000$  resolution, and to within a few percent the standard deviation of noise did not change.

### 8.1.2. 0 Voltage (Off) Test

The first tests had the function generator completely disconnected from the MEMS chip (and thus providing no voltage): the device being analyzed should remain completely stationary. Figure 49 shows a plot of voltage versus arbitrary time. It would be unlikely that any other circumstances could produce better results. Thus, this test was used as the base line for future tests, as it shows how both the XCorr and Yamahata degrade as conditions worsen. Figure 50 shows the ROIs selected for this test.

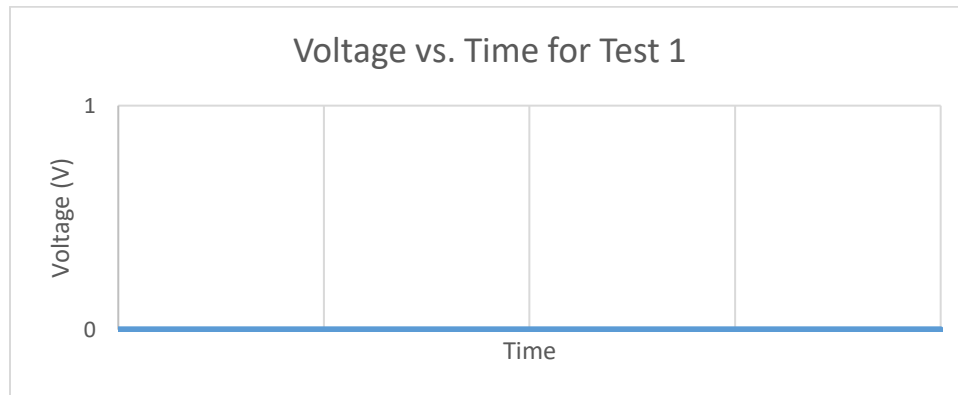


Figure 49: Stationary (Off) test: plot of voltage with respect to an arbitrary time

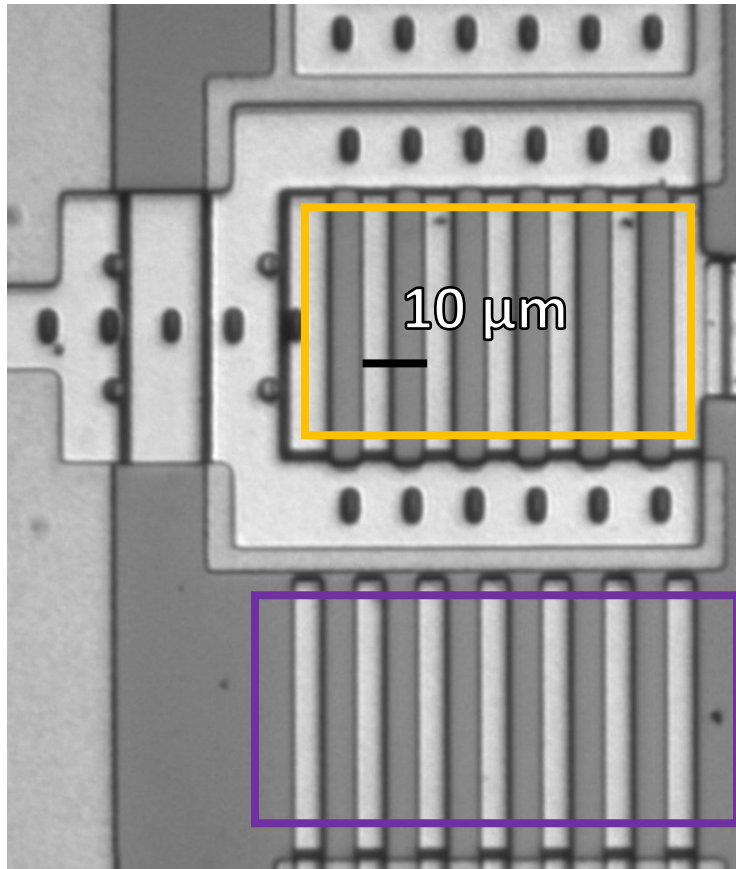


Figure 50: ROIs used in stationary test. Orange indicates moving ROI, and purple indicates reference ROI.

This test used the largest ROIs available, as shown in Figure 50. These ROIs were selected as they have a large number of periods and have the tallest periodic structures. Thus, the column averaging will eliminate as much noise as possible. Both of these conditions have been established as being important to the performance of the Yamahata algorithm.

Figure 51 compares the Xcorr and Yamahata results for one such stationary test and shows the measurements from the same 50 images analyzed by both the XCorr and the Yamahata algorithms. In both cases, the first point is measured as 0, as this is the initializing image. All subsequent measurements are compared to the first image, which is used as the reference point. The XCorr results proved to be consistently better than the Yamahata. In Figure 51 the standard deviation for Yamahata was  $\sigma = 1.21$  nm, while for Xcorr it was  $\sigma = 0.25$  nm.

Figure 52 shows a summary of the  $\sigma$  for Xcorr and Yamahata for 5 different stationary tests. The average result for Yamahata was  $\sigma = 1.21$  nm while for XCorr was  $\sigma = 0.27$  nm. The result is more consistent, and the precision of the measurement was on average 4.5 times better than the Yamahata performance. For a pixel size of 163 nm, the XCorr is capable of measuring  $\sim 1/600$  of a pixel vs  $\sim 1/120$  pixel for Yamahata.

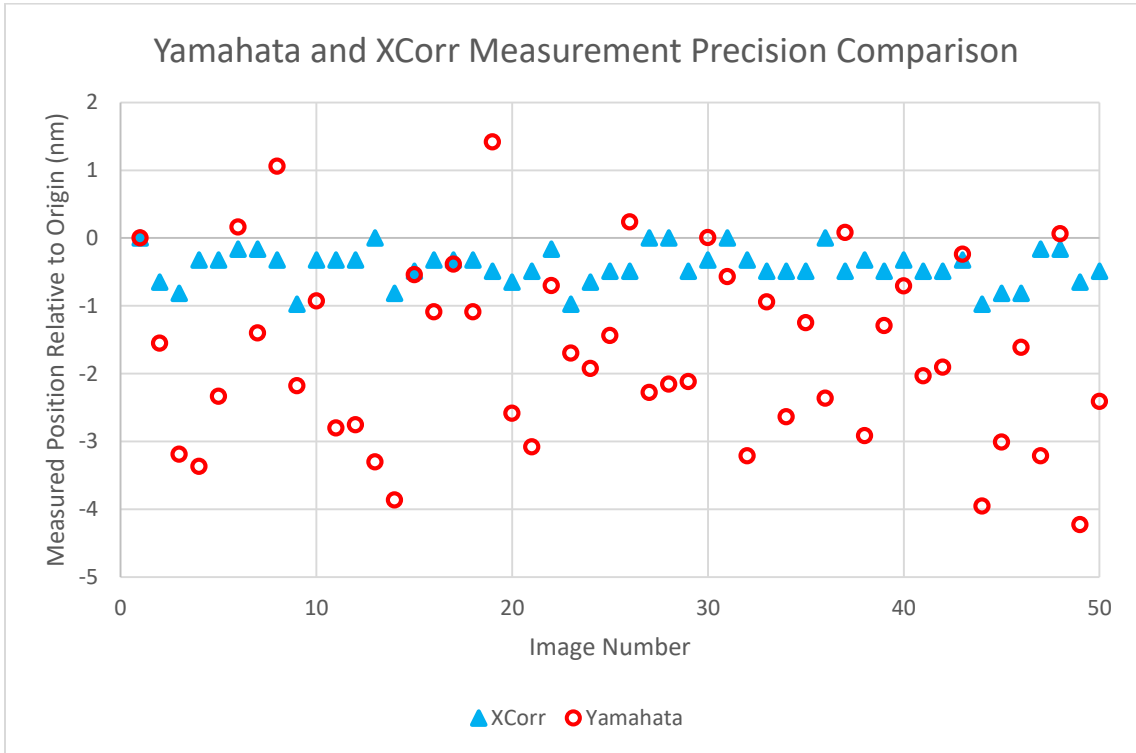


Figure 51: Results of Test 5 (Off), showing the position measurements of 50 images for XCorr and Yamahata.

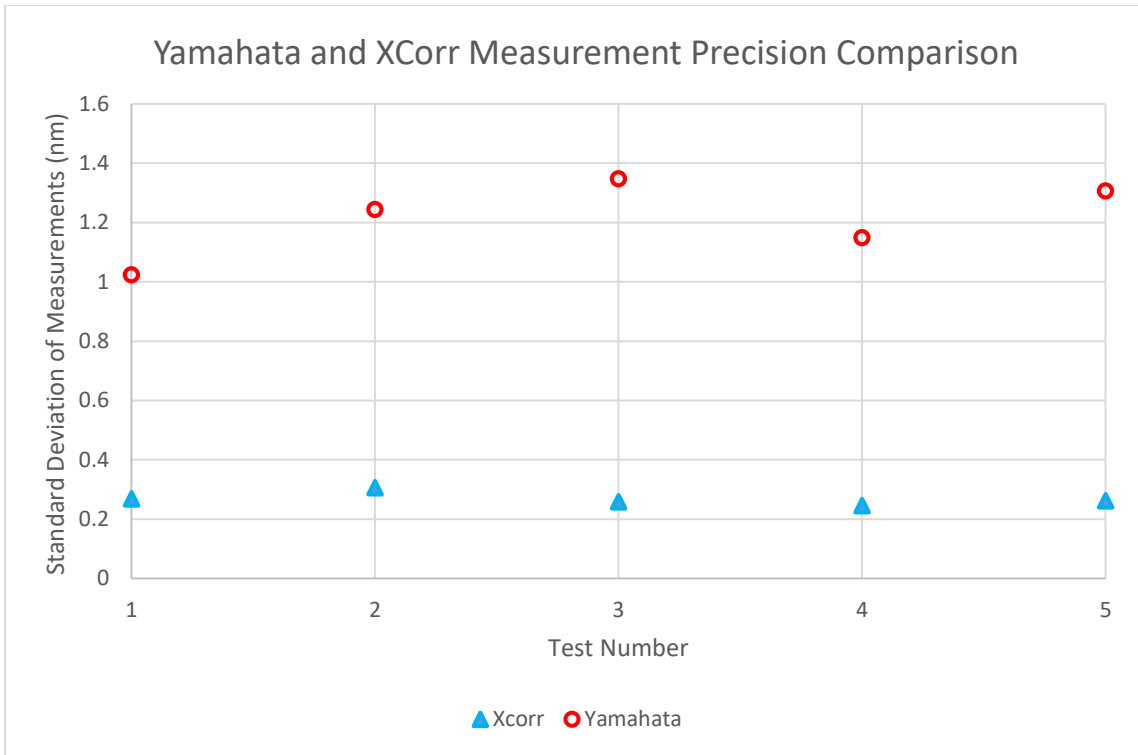


Figure 52: Standard deviation of measurements for the five tests. Average value for XCorr was 0.27 nm, and average value for Yamahata was 1.21 nm.



### 8.1.3. 1 Volt (On) Test

A second iteration on the test was performed, this time with a constant voltage of 1 V applied to the system. Figure 53 shows a plot of voltage versus arbitrary time. Comparison to the 0 V (Off) case will help to determine if there are any sources of error caused by variations in motion of the thermal actuators as a result of an applied voltage. Possible sources of motion noise include current/voltage variations and convective or thermal effects.

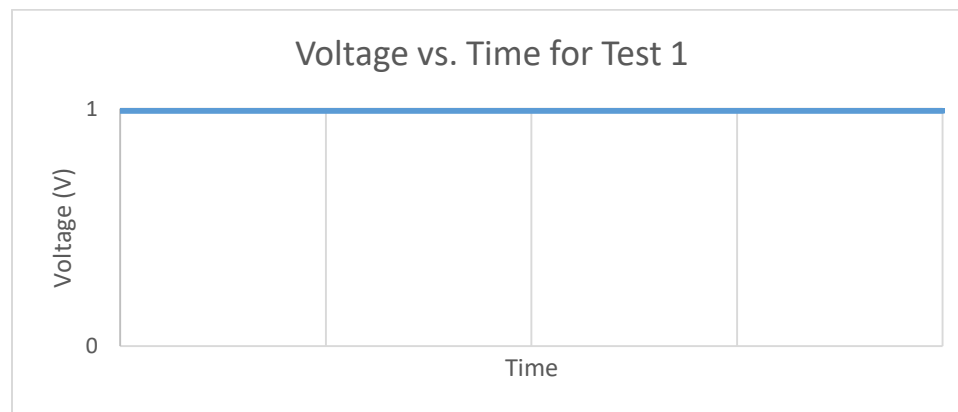


Figure 53: 1 Volt (On) test: plot of voltage with respect to an arbitrary time

In theory, the ROI should remain stationary, resulting in the measurements having a standard deviation similar to the previous tests. However, Figure 54 shows that the standard deviation of the measurements in the XCorr increased by approximately 33%, from 0.27 nm to 0.36 nm. The Yamahata did not see a notable drop in performance, with the standard deviation changing from 1.21 nm to 1.23 nm.

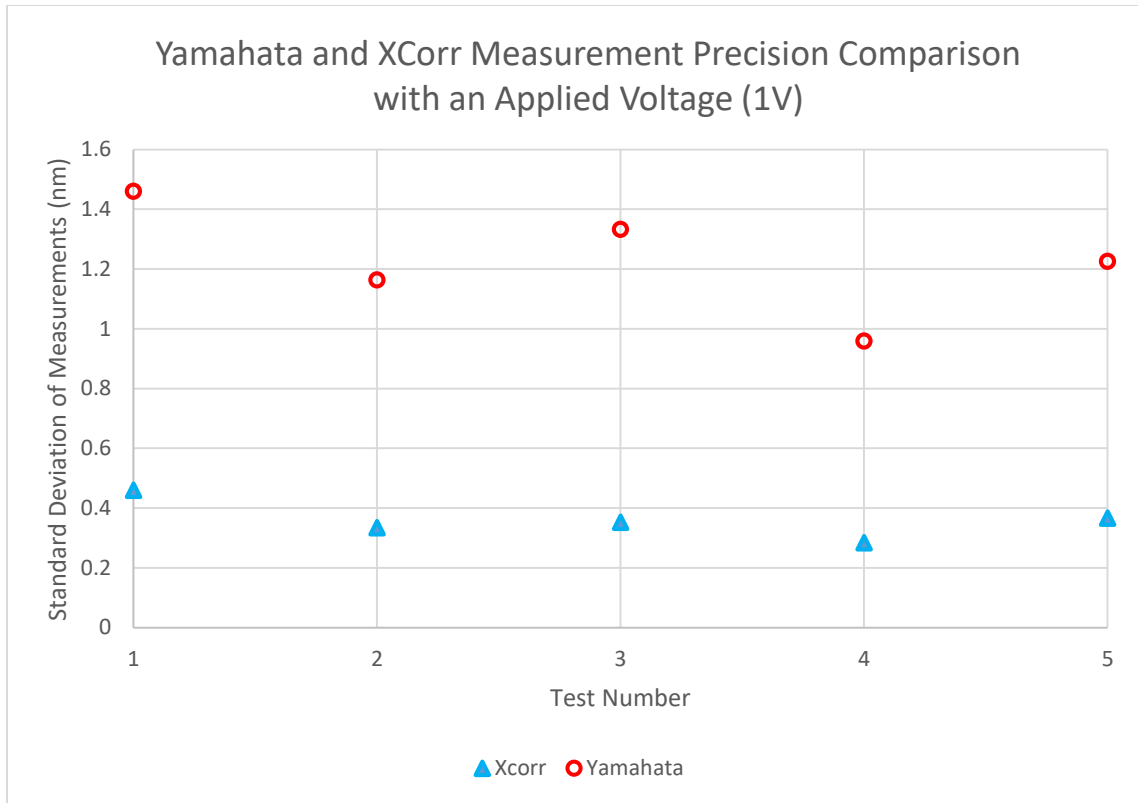


Figure 54: 1 Volt (On) test: standard deviations  $\sigma$  of five tests. Average  $\sigma$  for XCorr was 0.36 nm, and average  $\sigma$  for Yamahata was 1.23 nm.

The most likely cause of this is that the application of a voltage to the thermal actuators will present a physical motion noise in addition to the optical noise present in the camera. Given that the optical noise of the XCorr algorithm had been 0.27 nm without an applied voltage, and was 0.36 nm with an applied voltage, a rough estimate of the noise can be found.

Assuming that:

$$\sigma_{optical}^2 + \sigma_{physical}^2 = \sigma_{total}^2 \quad (16)$$

It can then be assumed that:

$$\sigma_{physical} = \sqrt{\sigma_{total}^2 - \sigma_{optical}^2}$$

$$\sigma_{physical} = \sqrt{.36^2 - .27^2}$$

$$\sigma_{physical} = \sqrt{.36^2 - .27^2}$$

$$\sigma_{physical} = 0.24 \text{ nm}$$

The same equation can then be applied to the Yamahata results to find what the standard deviation of the results with an applied voltage should be.

$$\sigma_{total} = \sqrt{\sigma_{physical}^2 + \sigma_{optical}^2}$$

$$\sigma_{total} = \sqrt{.24^2 + 1.21^2}$$

$$\sigma_{total} = 1.24 \text{ nm}$$

Given that this is a very minor difference in the expected result, it seems reasonable to assume that this physical motion noise may have been a large part of the difference in results observed in the XCorr, while not being observable in the Yamahata results. The physical motion noise would likely be caused by variations in current/voltage, or by variations in the convective heat losses experienced by the thermal actuators, which could be difficult to control. This error may not occur in all cases, and would likely depend heavily on conditions in the lab during testing. Further testing may be desired to confirm these results. With the Yamahata algorithm, this additional error had never previously been observed in the Dalhousie MEMS lab.

#### 8.1.4. 1.0 to 1.1 Volt STEP Test

For step detection, the objective was to determine if measurements can be accurately made to within the scope of the noise. In Chapter 3, a parabolic formula for motion of thermal actuators was found. Using the parabolic fit, at 1V the slope is approximately 250 nm/V. It predicts that for a 100 mv input the 1V-1.1V step should 25 nm and it predicts that for a 10mv input the 1V - 1.01V step should 2.5 nm

A 100 mV step test was run starting at 1.0 V, and stepping up to 1.1 V. Figure 55 plots the change in voltage in arbitrary time. This step was repeated three times, with ten pictures taken at each voltage, for 60 total pictures. The results of this test are plotted in Figure 56.

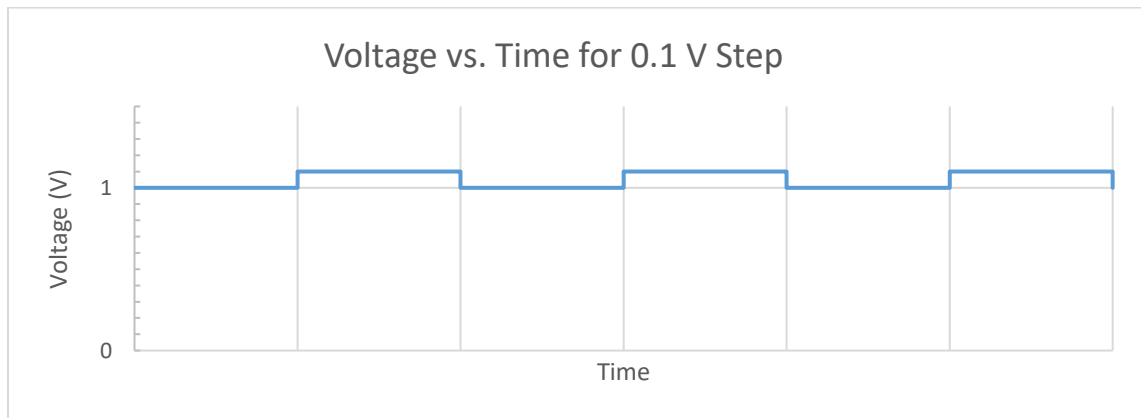


Figure 55: 100 mv step test: Plot of voltage with respect to an arbitrary time.

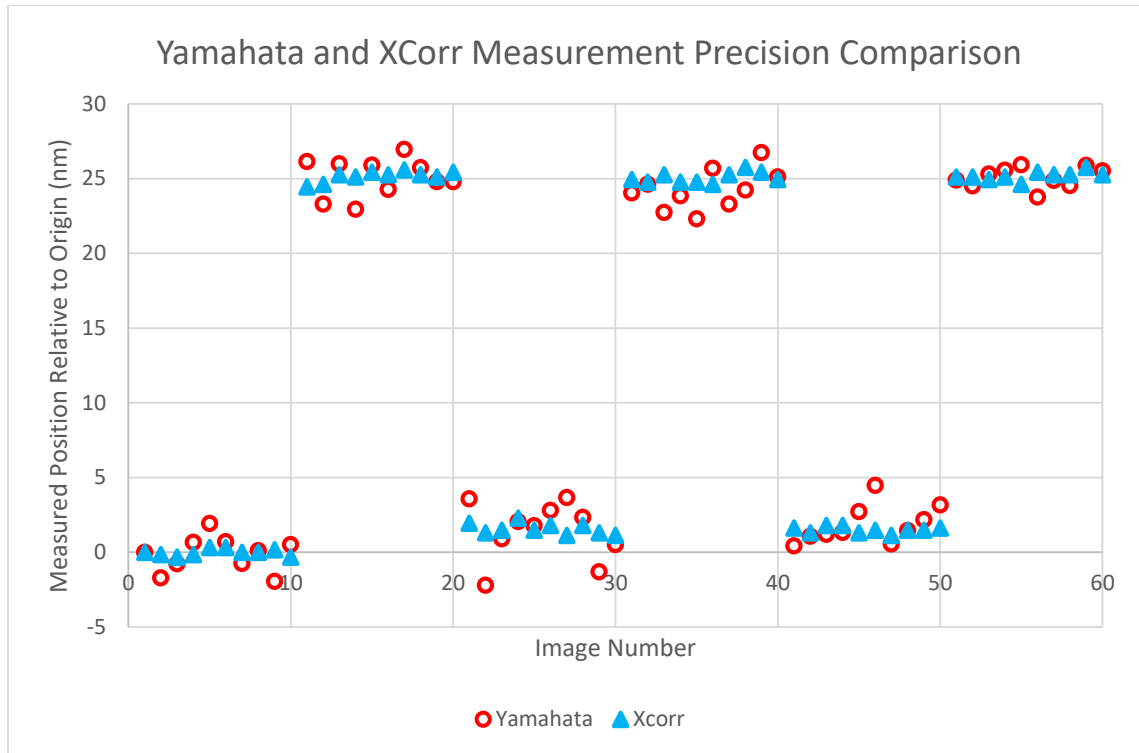


Figure 56: The XCorr and Yamahata results of a 25 nm step.

Both the Yamahata and XCorr algorithms show very clearly defined steps. The actuator does not return to its original position after the first step (images 11-20 in Figure 56), which is demonstrated by both algorithms. This may be due to the additional heat in the system from the 1.1 V step, or due to friction preventing the actuator from returning to the initial condition, though it is difficult to say for certain. However, subsequent steps appear to maintain a consistent position, and this was seen through multiple similar tests. In Figure 56, the average standard deviation at each step for the XCorr was  $\sigma = 0.31$  nm, while the Yamahata algorithm had a standard deviation of 1.29 nm on average.

#### 8.1.5. 1.00 to 1.01 Volt STEP Test

It was desired to see if the XCorr and the Yamahata algorithms would be able to detect smaller changes in displacement. By reducing the step increase from 100 mV to 10 mV, a displacement of 2.5 nm should be observed. Figure 57 plots the voltage change with respect to time.

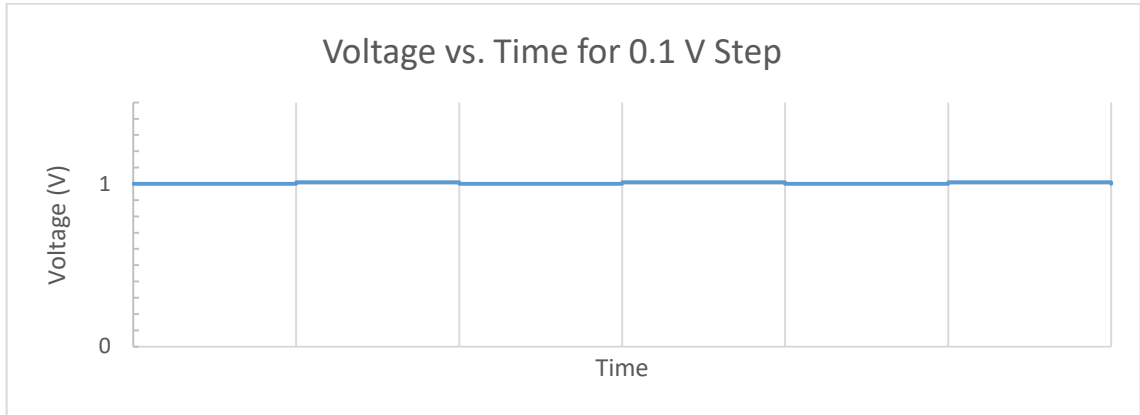


Figure 57: Plot of voltage with respect to an arbitrary time.

The 2.5 nm step would be approximately 1/65 of a pixel of motion, and the motion is completely imperceptible to a person looking at the images. Figure 58 plots the results of this test, as found by both the XCorr and Yamahata algorithms.

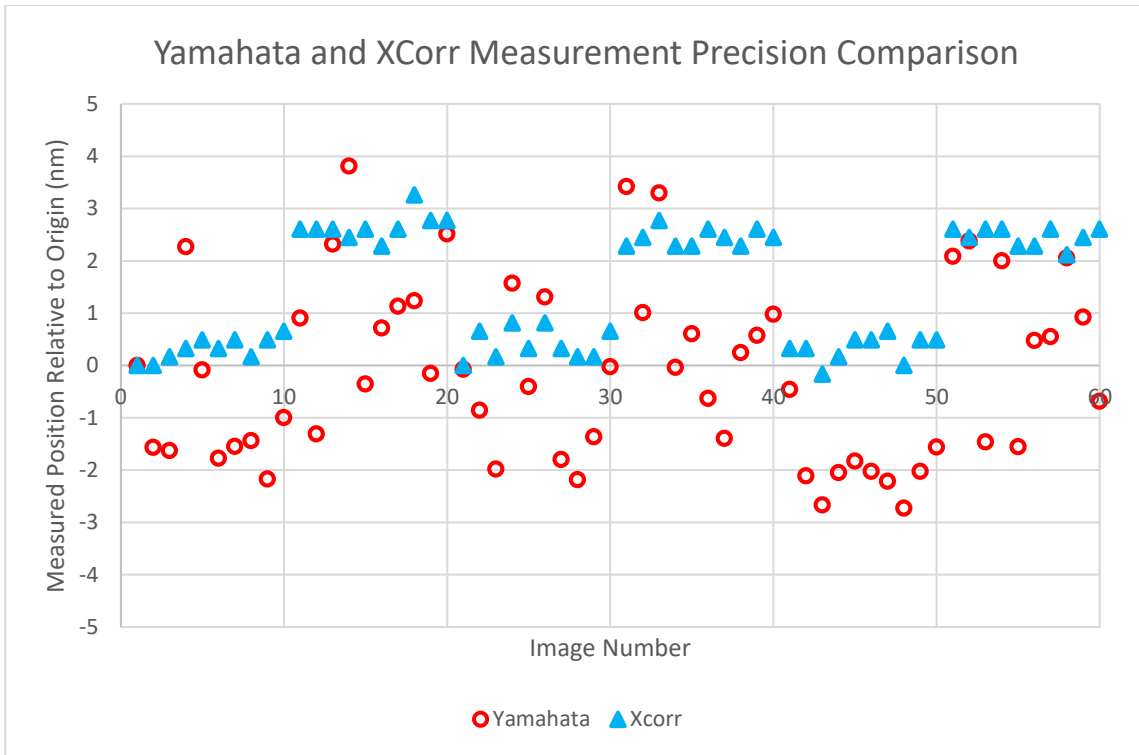


Figure 58: XCorr and Yamahata results of a 2.5 nm step.

The motion and the noise are difficult to distinguish from the Yamahata algorithm data. Figure 59 shows the average value and standard deviation of each step. The XCorr algorithm, on the other hand, can clearly identify the steps, and the average values are all within 0.5 nm of the expected values. As with the previous case, the actuator doesn't appear to return to the exact initial value, but the subsequent steps are consistent. Figure 60 shows just the results of the XCorr test, and Table 8 includes the average measurement at each step, as well as the standard deviation for these results.

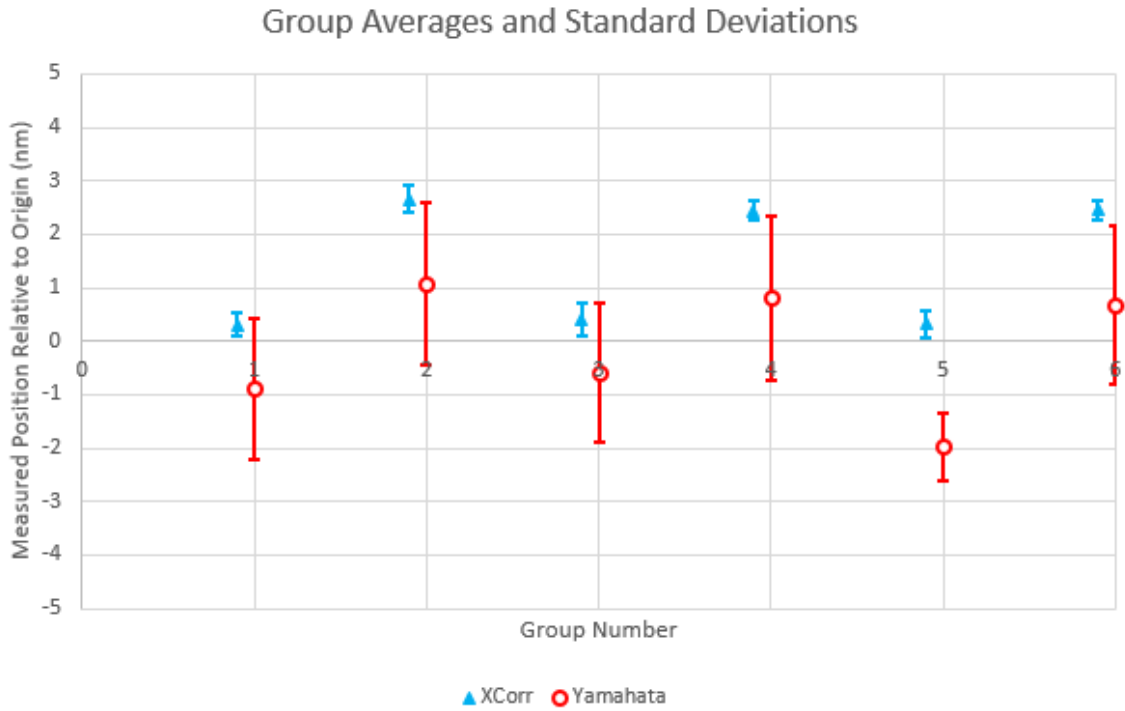


Figure 59: XCorr and Yamahata grouped averages for 2.5 nm step.

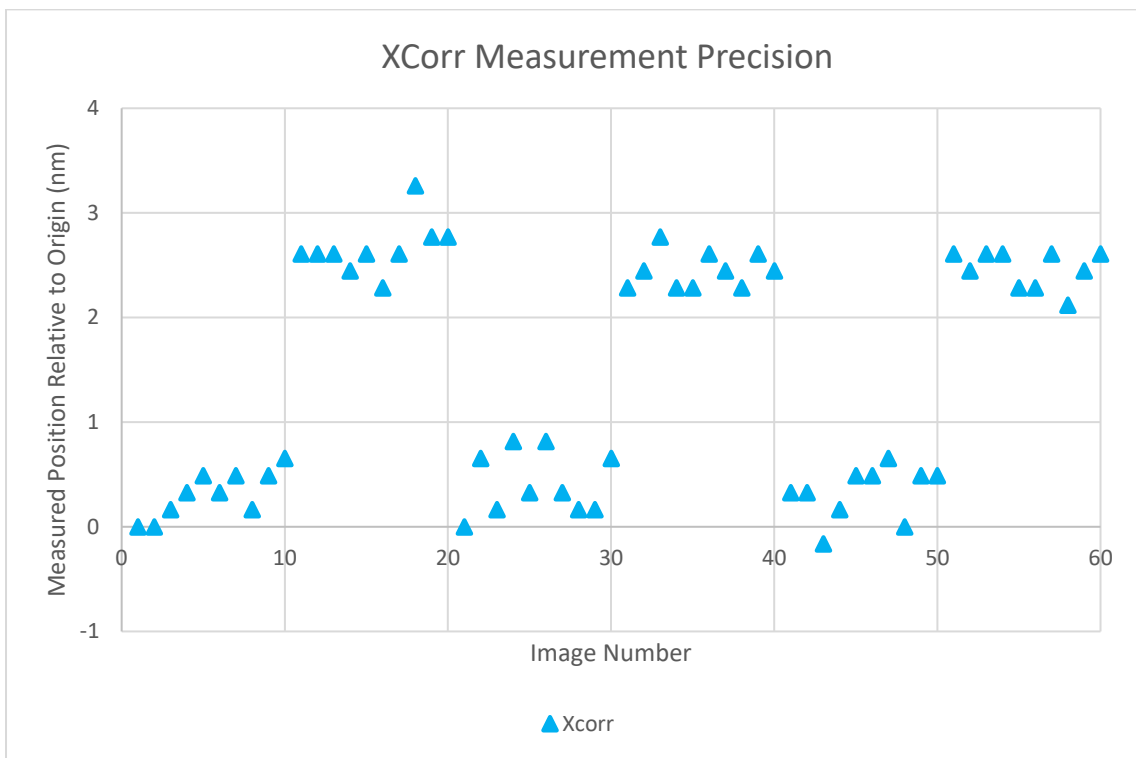


Figure 60: 10 mv step test: XCorr results for a 2.5 nm step.



Table 8: The average measurement and standard deviation at each step in the 1.00 – 1.01 mV test as measured by the XCorr.

<b>Step Number</b>	<b>Average Measurement (nm)</b>	<b>Standard Deviation (nm)</b>	<b>Average Step (nm)</b>
1	0.31	0.22	2.35
2	2.66	0.26	
3	0.41	0.30	2.05
4	2.45	0.17	
5	0.33	0.25	2.13
6	2.46	0.18	

## 8.2. Measurement Noise

With the data from the previous tests, it is possible to determine how much noise is present within the system. The noise can then be compared to values from Chapter 6.2 in order to determine if the model is accurately representing real world results. The easiest way to find a value of noise is to select a region in which there are only minimal changes in the shade of the image and determine how much noise is present in this region. As such, the ROI indicated in Figure 61 was selected for this test.

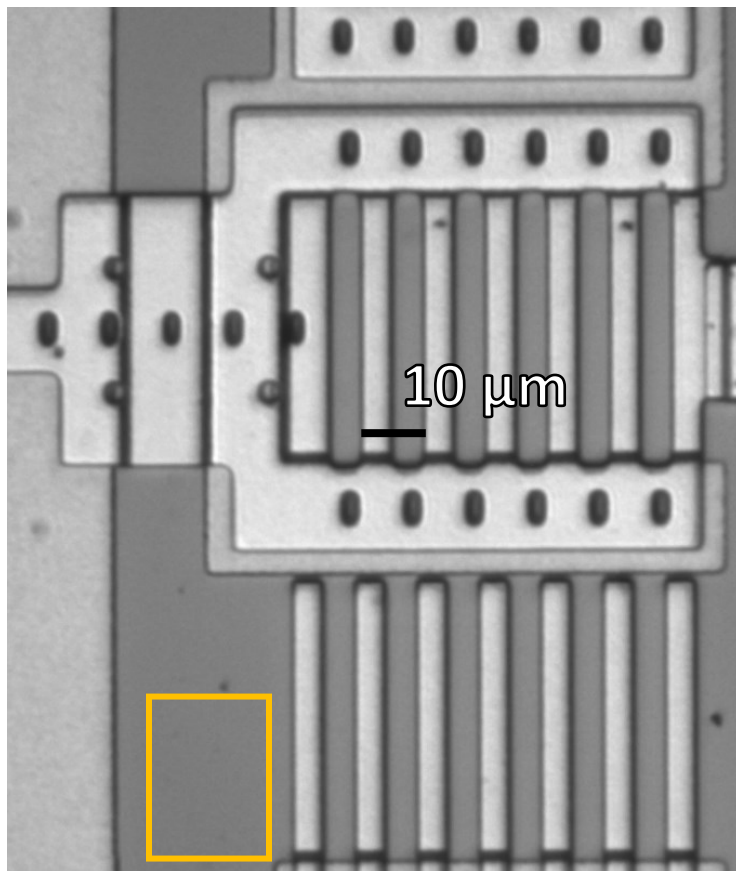


Figure 61: ROI (in orange) used in order to evaluate the noise in the data.

This ROI was column averaged and the offset and slope removed. The resulting profile is plotted in Figure 62 and this will be interpreted as the amount of noise present in the system. This plot is converted to a percentage of the 8-bit intensity values.

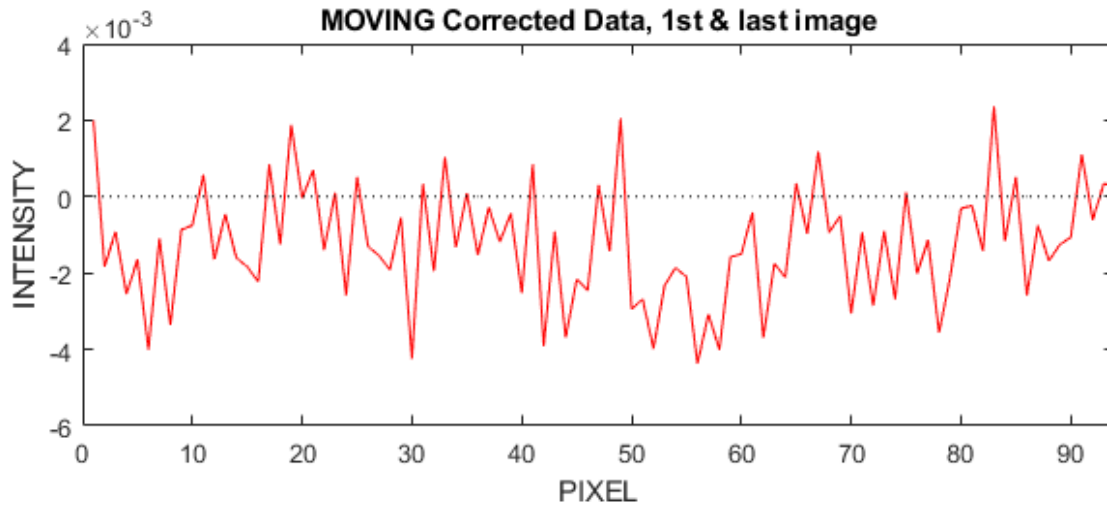


Figure 62: Adjusted intensity column averaged values of the ROI in Figure 61.

In this case, the maximum value within the range was found to be 0.25%, and the minimum was -0.45%. This represents 0.70% of the total range, which is smaller than any of the noise tests which had been conducted previously in the simulation in Chapter 2.5. Other sample ROIs provided similar range values.

The Chapter 2 simulations were re-run with a noise value of 0.0070. The standard deviation of the simulated measurements was found to be 0.002 pixels, which would be equivalent to 0.32 nm for our 163 nm pixels. This is slightly larger than the 0.27 nm average value calculated under the ideal circumstances of the 0 V (Off) test in Chapter 6.1. However, the two test setups are not quite the same, as the real images included 6 periods, and the simulation used only 4.

### 8.3. Impact of Colour

As was discussed in Chapter 2.4, the images used in these experiments are RGB images. Pixels in each image are composed of red, green, and blue components, producing the colours of the image. Figure 63 shows four different versions of the same image, the first being the RGB image, and the other three being the three component colours individually. For the purposes of the image processing completed in these experiments, these are typically converted to a greyscale image using standard conversion factors.

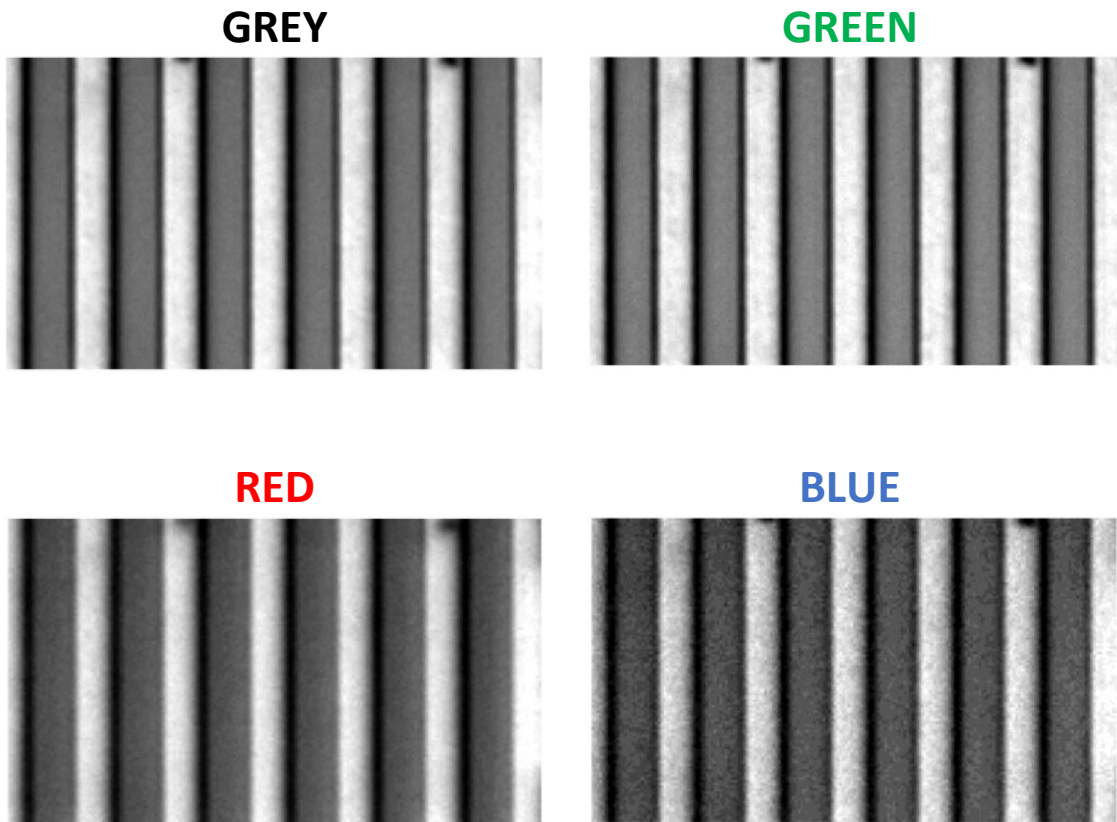


Figure 63: A sample image (top left) and its red, green, and blue components.

From the above image it may be noted that the grey and green image appears to provide the sharpest result. Red is less sharp, and blue is far noisier. Part of the reason for this is the Bayer filter on the camera image sensor. A Bayer filter is the grid which collects light in order to produce an RGB image and is used in most basic image sensors. The grid is composed of a repeating pattern of four squares, two of which are green, one of which is red, and one of which is blue. These four squares are arranged in a 2x2 pattern, with the two green squares placed at opposite corners to one another. Figure 64 shows a simple Bayer filter with the described pattern in the top left corner. This pattern is repeated in both the x and y direction.

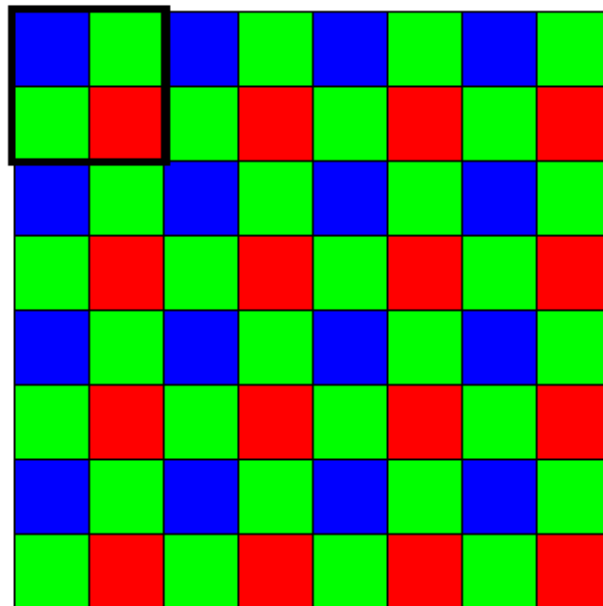


Figure 64: A sample Bayer filter, with the repeating pattern emphasized in the top left corner.

This led to an investigation into the impact of colours, and how they are modified to produce the greyscale images being investigated. Figure 65 plots the results of a standard greyscale measurement of 1 - 1.01V 2.5 nm steps for four different colour channels of the same ROI: grey, green, red, and blue. The green and greyscale produce similar results. The red shows a notable reduction in quality, however the steps are still distinguishable. When measuring using the blue light, the steps are indistinguishable. Table 9 indicates the average standard deviation for the four color tests.

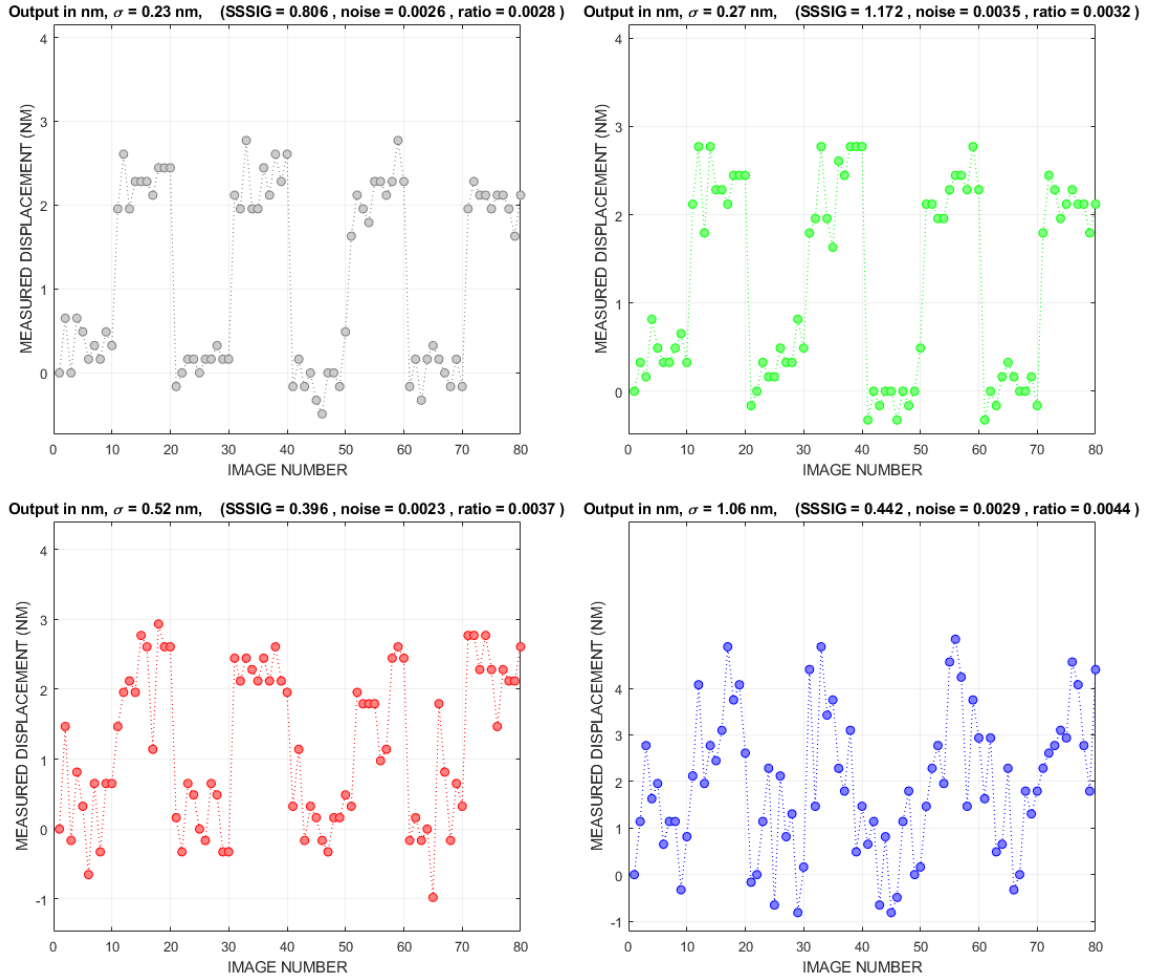


Figure 65: Results of greyscale, green, red, and blue color channels (1 – 1.01V 2.5 m steps).

Table 9: Results of measurements taken using different colour scales.

Colour	Standard Deviation (nm)
Greyscale	0.27
Green	0.29
Red	0.57
Blue	1.09

## 8.4. Impact of Number of Periods, Bar vs. Dot, Size in Yamahata / XCorr

### 8.4.1. Why is this Important?

The largest limitation in the existing Yamahata algorithm is the requirement for periodic structures in order to obtain precise results. The following tests will evaluate how both the existing Yamahata algorithm and the new XCorr algorithm are affected by changes to the ROI. This will show the ability to utilize these algorithms in a variety of circumstances, some of which may not have defined periodic structures.

### 8.4.2. Test Setup

These tests will use the same sets of images as Section 7.1, however the ROI will be changed from test to test. This will be done in a couple of ways. First, the number of bars used in both the moving and reference ROI will be reduced in order to determine the impact of the number of periods being processed. Second, the height of the ROI will be changed in order to determine the impact of column height. Third, the structures used for the ROI will be changed from a long bar to an oval dimple in order to determine the impact of structure shape. Finally, non-periodic conditions will be tested.

#### 8.4.3. Bar Number

The following tests evaluate the differences present when altering the number of periodic structures used in testing. Figure 66 shows the initial ROI (orange solid line) that was used, which covers 6 periods. Successively narrower ROIs were then selected covering 5, 4, 3, 2 and 1 period(s). The purple dashed line indicates the smallest ROI tested (1 period). A similar change in the ROI was also applied to the reference. Figure 67 shows each of these six ROIs.

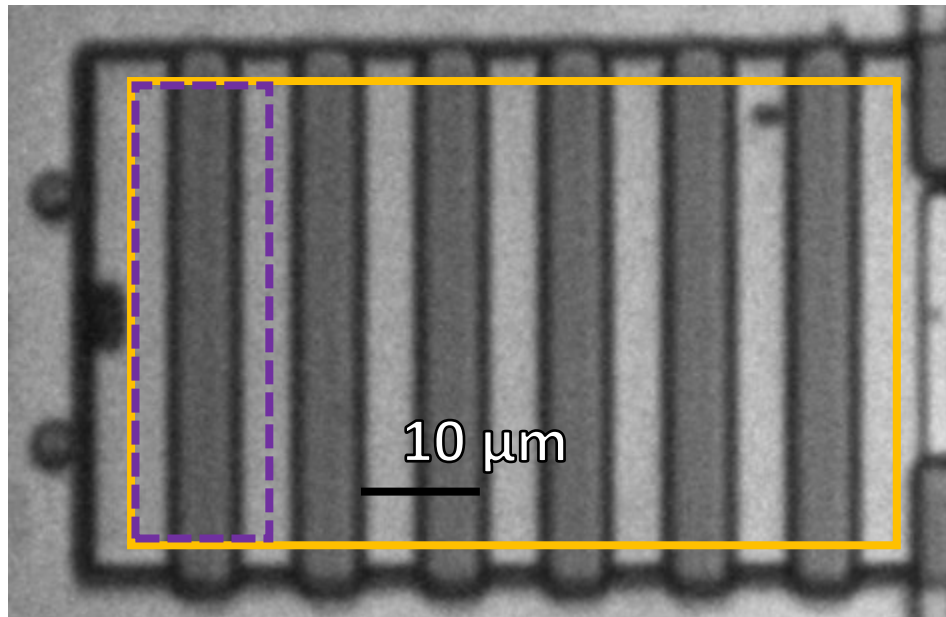


Figure 66: The initial ROI (orange), and the one period ROI (purple, dashed line).



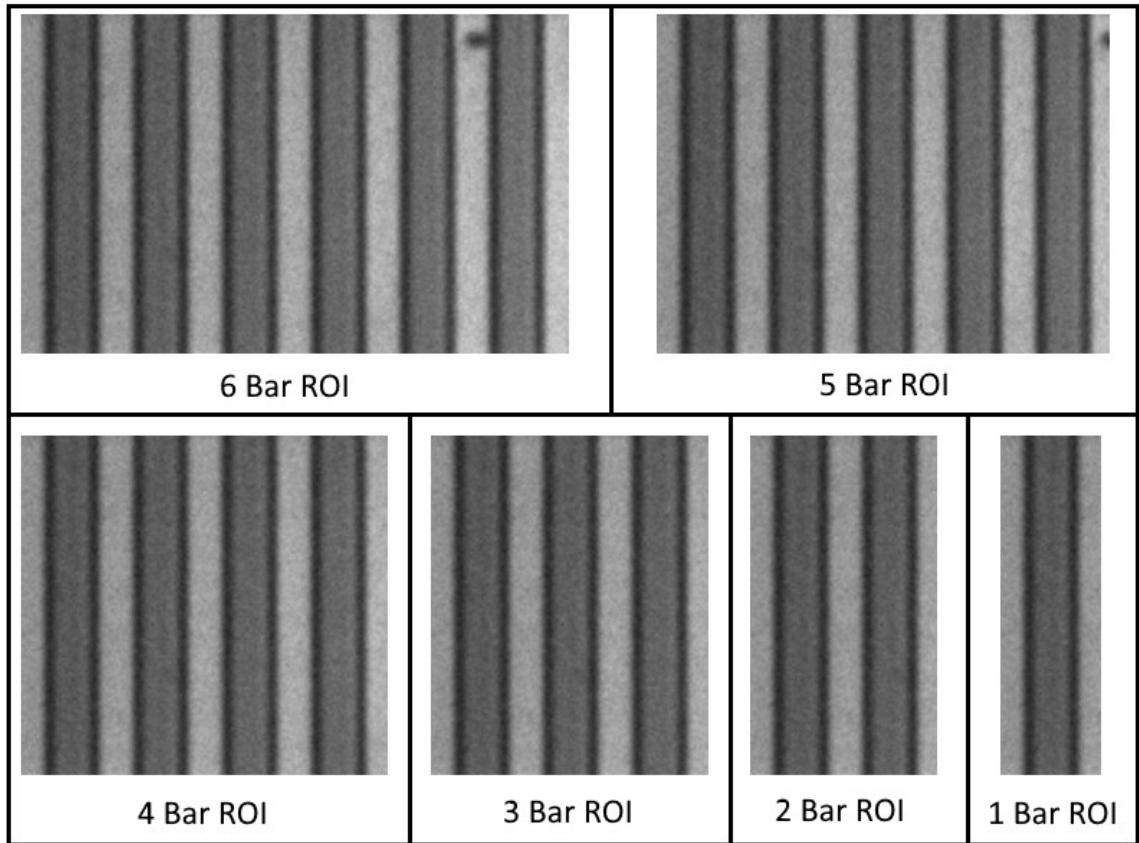


Figure 67: ROIs used when evaluating impact of period number on measurements precision.

The results of the Yamahata and XCorr tests are plotted in Figure 68. Figure 69 presents a zoomed in version, including all measurements except for the Yamahata results with 1 period. Additionally, Figure 69 includes power function fits of the data. In the case of the Yamahata results, the data for the 1 Bar ROI is omitted. Note that the Xcorr power fit of -0.552 is close to the  $x^{-0.5}$  predicted, assuming that SSSIG is proportional to ROI width. The exact values, as well as a comparison of the ratios between the two tests, can be found in Table 10.

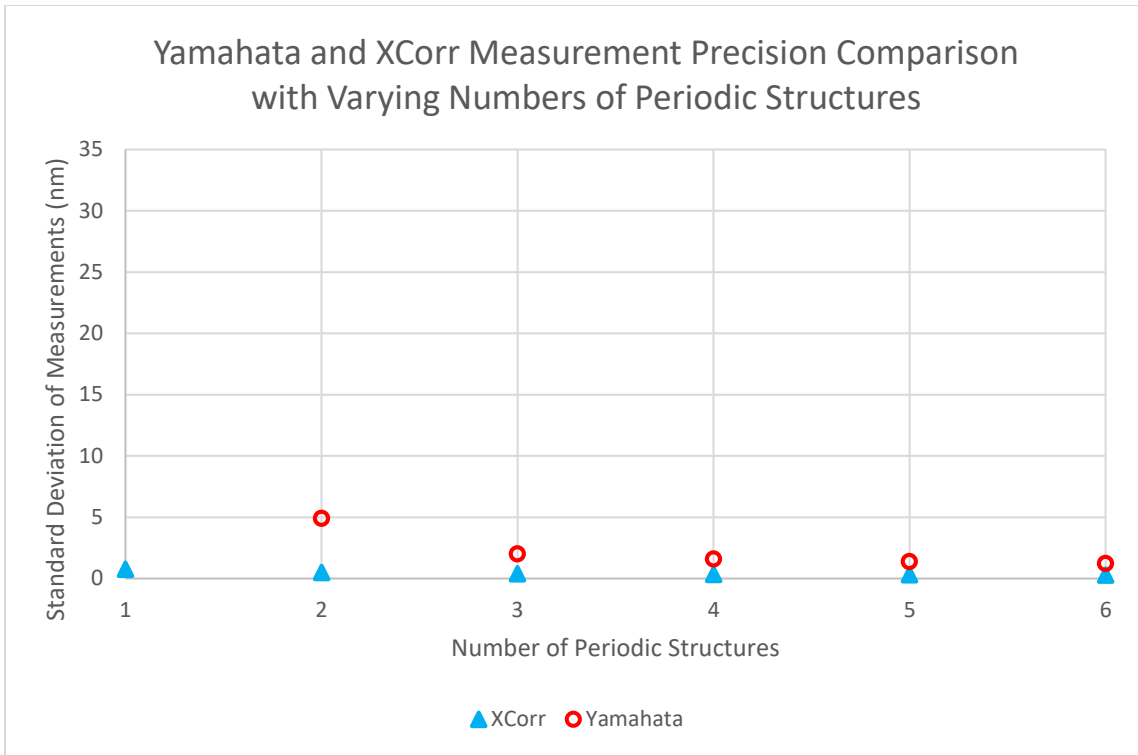


Figure 68: The results for 1-6 periods of Yamahata and Xcorr.

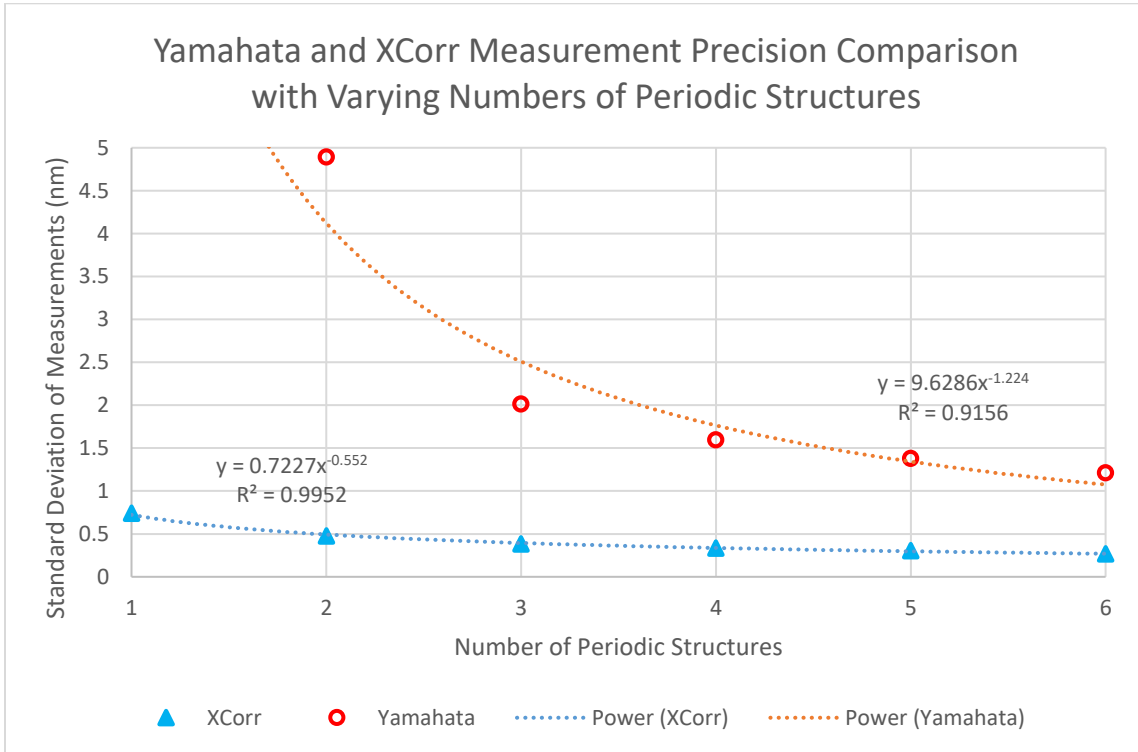


Figure 69: Zoom in of Figure 68, for the purposes of better visualizing the results.

Table 10: Measured standard deviation for XCorr and Yamahata algorithms as number of periods is varied.

Periods	XCorr $\sigma$ (nm)	Yamahata $\sigma$ (nm)	Yamahata/XCorr
6	0.27	1.21	4.5
5	0.31	1.38	4.5
4	0.34	1.60	4.7
3	0.38	2.01	5.2
2	0.48	4.9	10.3
1	0.74	34.3	46

The precision of both the Yamahata and XCorr algorithms level off after 5-6 periods, indicating that additional periods would only result in marginally improved results. Decreasing the number of periodic structures, degrades the performance of both XCorr and Yamahata, but Yamahata is affected far more. From 6 periods to 1, the XCorr becomes about 2.7x worse (0.26 to 0.74 nm) while the Yamahata becomes 27x worse (1.2 to 32 nm). Compared to Yamahata, XCorr will produce progressively better results as the number of periods is decreased. With only 1 period, the XCorr still produces a standard deviation that is lower than Yamahata at 6 periods. XCorr should be able to be used in a wide variety of designs as it requires only a single period to produce sub-nm measurements.

Finally, the XCorr was tested for 1/2 period. The Yamahata is unable to process this data, as it requires at least a full period of data in order to evaluate a set of data. The XCorr requires that the data start and end at values of 0, however by accounting for the slope of the signal from start to finish, both of the end points can be set to 0. Figure 70 shows the ROI selected from the moving device. Figure 71 shows the raw and slope corrected profiles. In this case, a slope is added to the raw data such that the beginning and ending of the ROI will have values of 0, as the ROI begins in a light region, and ends in a dark region. Figure 72 shows measurements taken for the XCorr on a single edge, as well as the Yamahata using 1 periods. The XCorr average measurement showed  $\sigma = 1$

nm. This indicates that the XCorr can be used on any device with a linear edge, without periodicity.

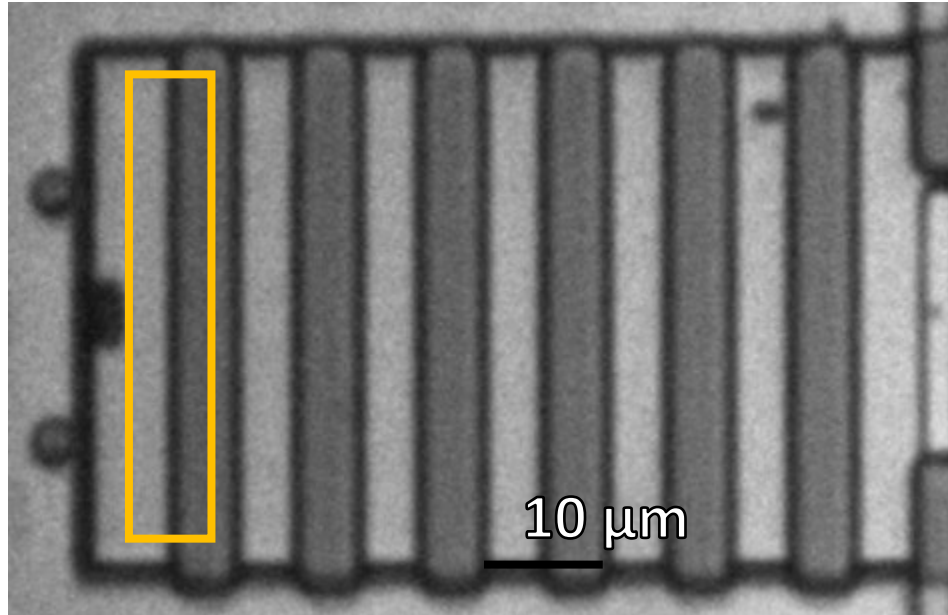


Figure 70: ROI selected for the “half period” tests.

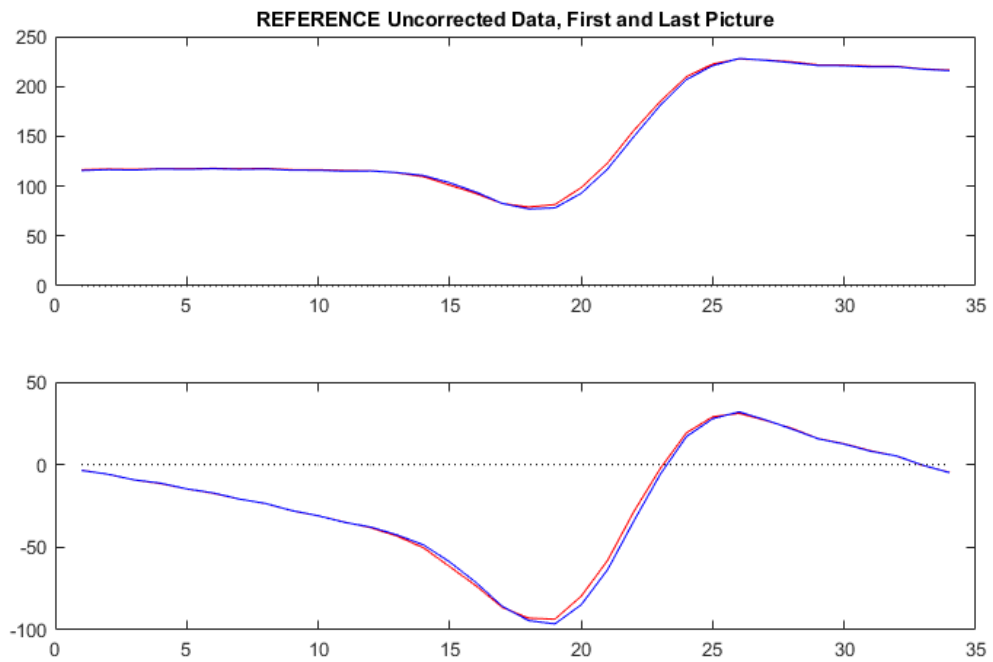


Figure 71: The raw (above) and adjusted (below) column averages of the first (red) and last (blue) images used in an edge detection test (2 nm motion).

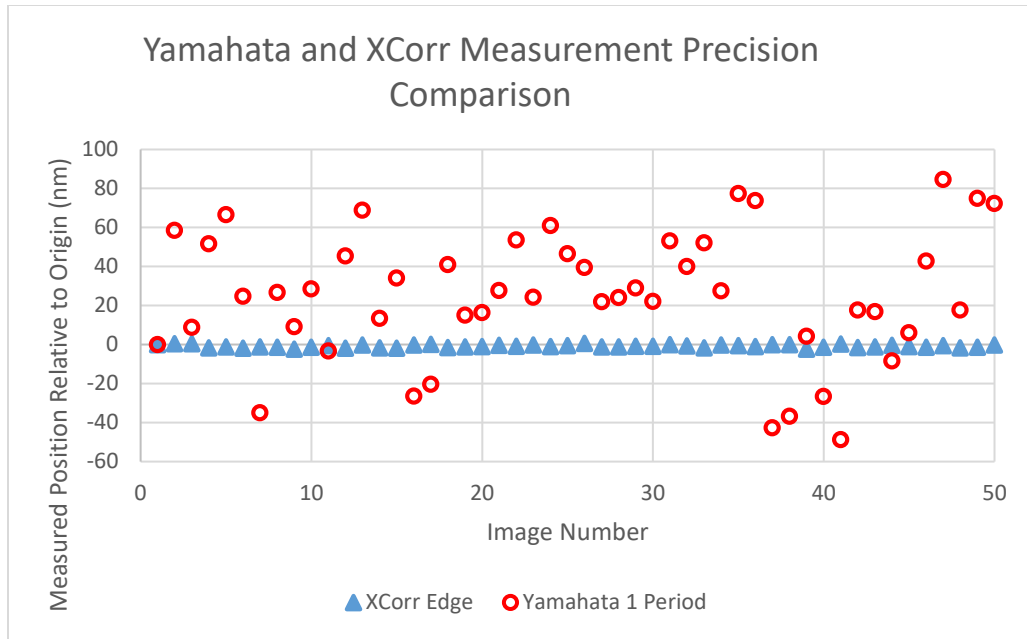


Figure 72: XCorr 1/2 period ROI vs. Yamahata with a 1 period ROI.

#### 8.4.4. ROI Height

Decreasing the ROI height of the bars will cause an increase in noise, as fewer pixels are averaged. Figure 73 shows the initial ROI (orange solid line) that was used, which covers the full bar height of 250 pixels. Successively shorter ROIs were selected for heights of 150, 100, 50, 25, and 15 pixels (purple dotted line). Figure 74 shows all 6 of the ROIs used for this test. Similar changes in the ROI were also applied to the reference. Figure 75 plots the standard deviation of the measurement as the bar height is decreased for both the Yamahata and the XCorr including power fits. Note that the XCorr power fit of -0.427 is close to the  $x^{-0.5}$  predicted, assuming that the noise is proportional to the square root of ROI height.

Table 11 lists the data from both Yamahata and Xcorr tests.

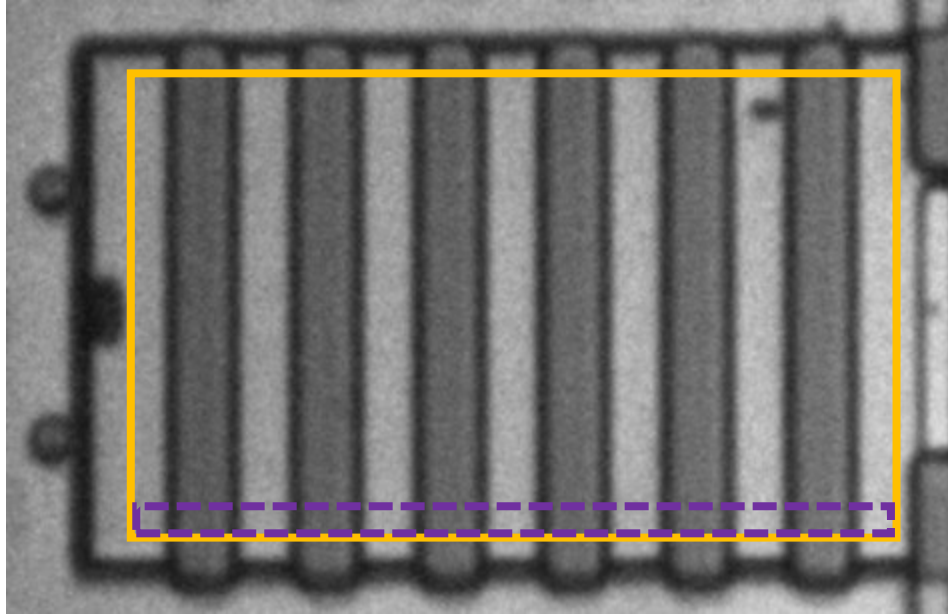


Figure 73: A comparison of the original ROI, and the smallest tested when evaluating the impact of ROI.

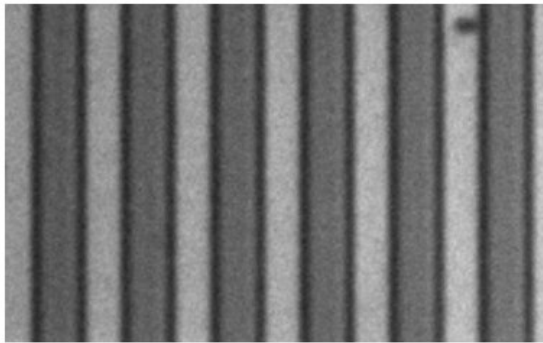
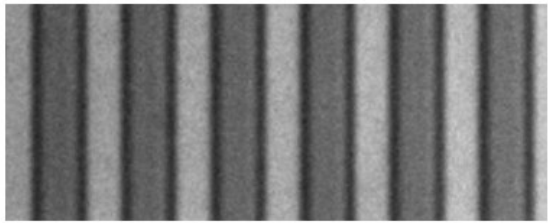
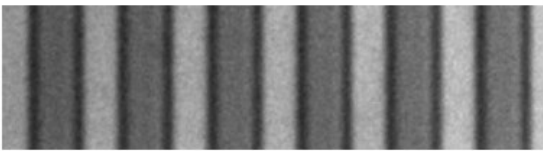
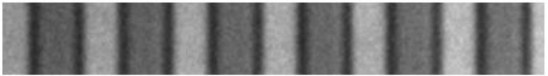


 <p>250 pixel height ROI</p>	 <p>150 pixel height ROI</p>
 <p>100 pixel height ROI</p>	 <p>50 pixel height ROI</p>
 <p>25 pixel height ROI</p>	 <p>15 pixel height ROI</p>

Figure 74: ROIs used in determining impact of bar height in measurement precision.

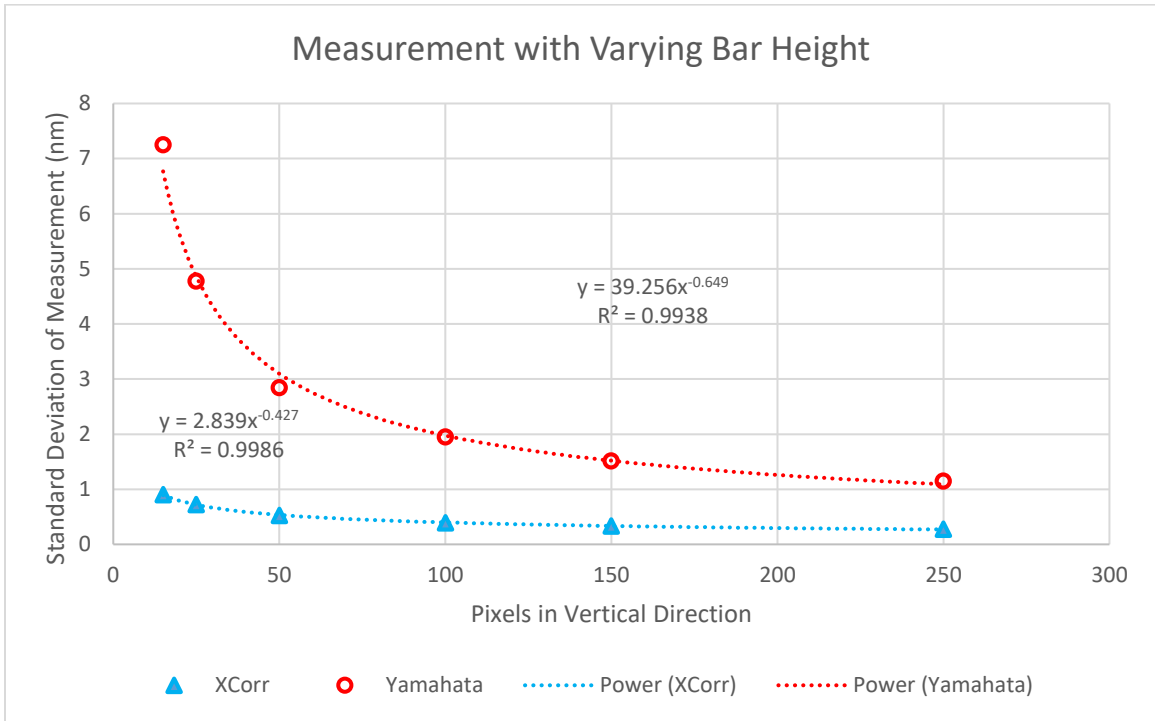


Figure 75: The results of the tests for XCorr and Yamahata, as well as their power fits.

Table 11: Measured standard deviations for the XCorr and Yamahata as the height of the ROIs were varied.

Height (Pixels)	XCorr $\sigma$ (nm)	Yamahata $\sigma$ (nm)	Yamahata/XCorr
250	0.27	1.1	4.2
150	0.34	1.5	4.5
100	0.39	1.9	5.0
50	0.52	2.8	5.4
25	0.72	4.8	6.6
15	0.91	7.3	8

Decreasing the bar height, degrades the performance of both XCorr and Yamahata: roughly as the  $1/\sqrt{N}$  where  $N$  is the number of pixels, but Yamahata is affected more. From 250 to 15 pixels, the XCorr becomes about 3.3x worse (0.27 → 0.91 nm) while the Yamahata becomes 6.2x worse (1.1 → 7.3 nm).

Both Yamahata and XCorr level off after ~200 pixels, additional pixels would only result in marginally better results. Even with a significantly reduced number of pixels the XCorr algorithm is consistently better than the results of the Yamahata. With only 15 pixels, the XCorr still produces a standard deviation that is better than Yamahata at 250 pixels.



#### 8.4.5. Dimples

A test was also conducted on a different type of periodic structure. Instead of repeated bars, repeated dimples in the device were used. The dimples are caused by small depressions on the structure. They also produce a sharp contrast. Figure 76 indicates the ROIs selected for the dimple tests. Figure 77 compares the column averaged profiles of the bars (left) and the dimples (right). The actual size of a dimple is about  $4 \times 6 \mu\text{m}$ . These are notably smaller than the bars, with a height of just 40 pixels, compared to about 250 pixels for the full bars. Table 12 presents the results of tests comparing the dimples to 40 pixel tall bars, using both Xcorr and Yamahata methods.

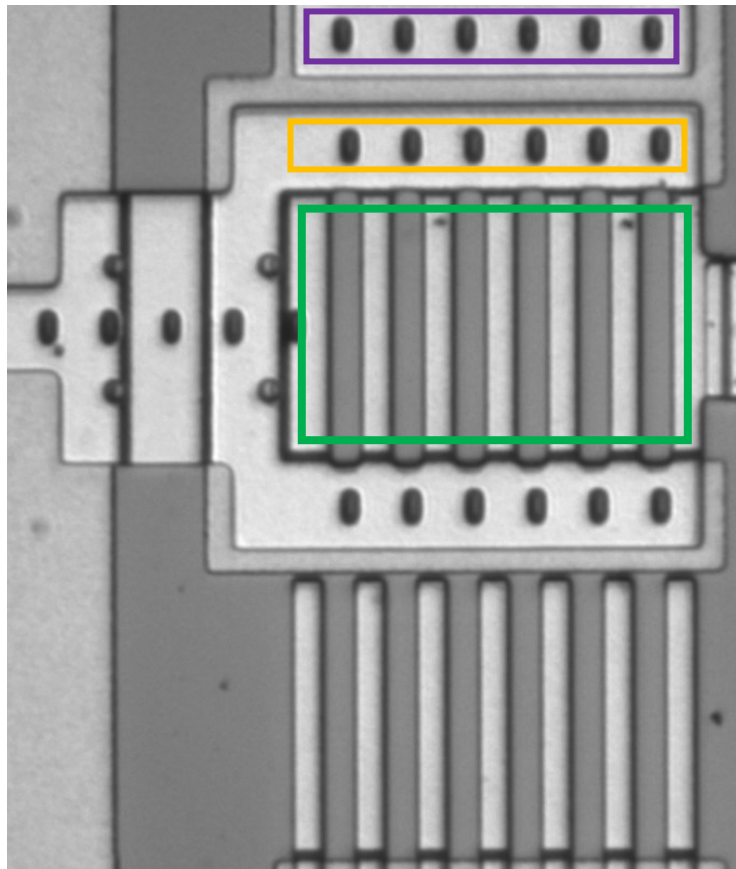


Figure 76: The moving ROI (orange) and the reference ROI (purple) for the dimples. The previously used ROI on the bars is also indicated (green)

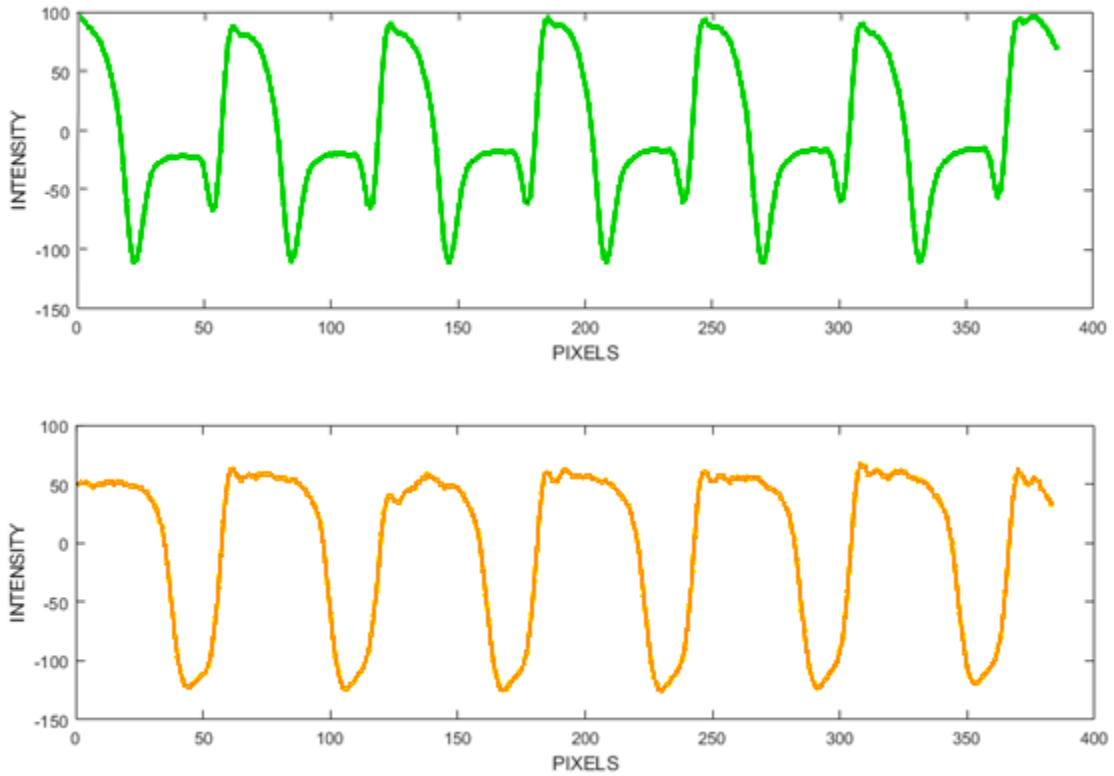


Figure 77: Column averages of the bars (above in green) and dimples (below in orange).

Table 12: Comparison of XCorr and Yamahata results for bars and dimples.

Test	Average $\sigma$ (nm)	Dimple/bar ratio
XCorr, Bars (40 px)	0.58	1.26
XCorr, Dimples	0.73	
Yamahata, Bars (40 px)	3.58	0.62
Yamahata, Dimples	2.19	

From Table 12 the XCorr results are noisier for the dimples vs. the bars, while Yamahata is the opposite. For the bars the XCorr would have an expected value of 0.58 nm, while the Yamahata would have an expected value of 3.58 nm, making the XCorr 6.1 times better than the Yamahata. In the case of the dimples however, they are only 3 times better. This suggests that the structure plays a notable role in how each process interprets the data, but the exact relationship is unclear. As for the XCorr, the SSSIG and

noise levels are ~10% worse for the dimples, but the difference in noise is larger than 10% would predict. As for the Yamahata the dimples having a more sinusoidal structure making the fundamental frequency more prominent may be beneficial.

#### 8.4.6. Non-Periodic Structures

As a result of the cross correlation algorithm used in the new displacement measurements, the XCorr does not require periodic structures. It can evaluate any structure with large contrasts. Figure 78 indicates the ROIs used in a non-periodic test. Figure 79 plots the column averages for both ROIs, with the moving ROI on the left and the reference ROI on the right. Neither of these structures has a consistent period, however both still have regions of high contrast. Table 13 compares the results for bar dimple and non-periodic structures, using both Xcorr and Yamahata methods.

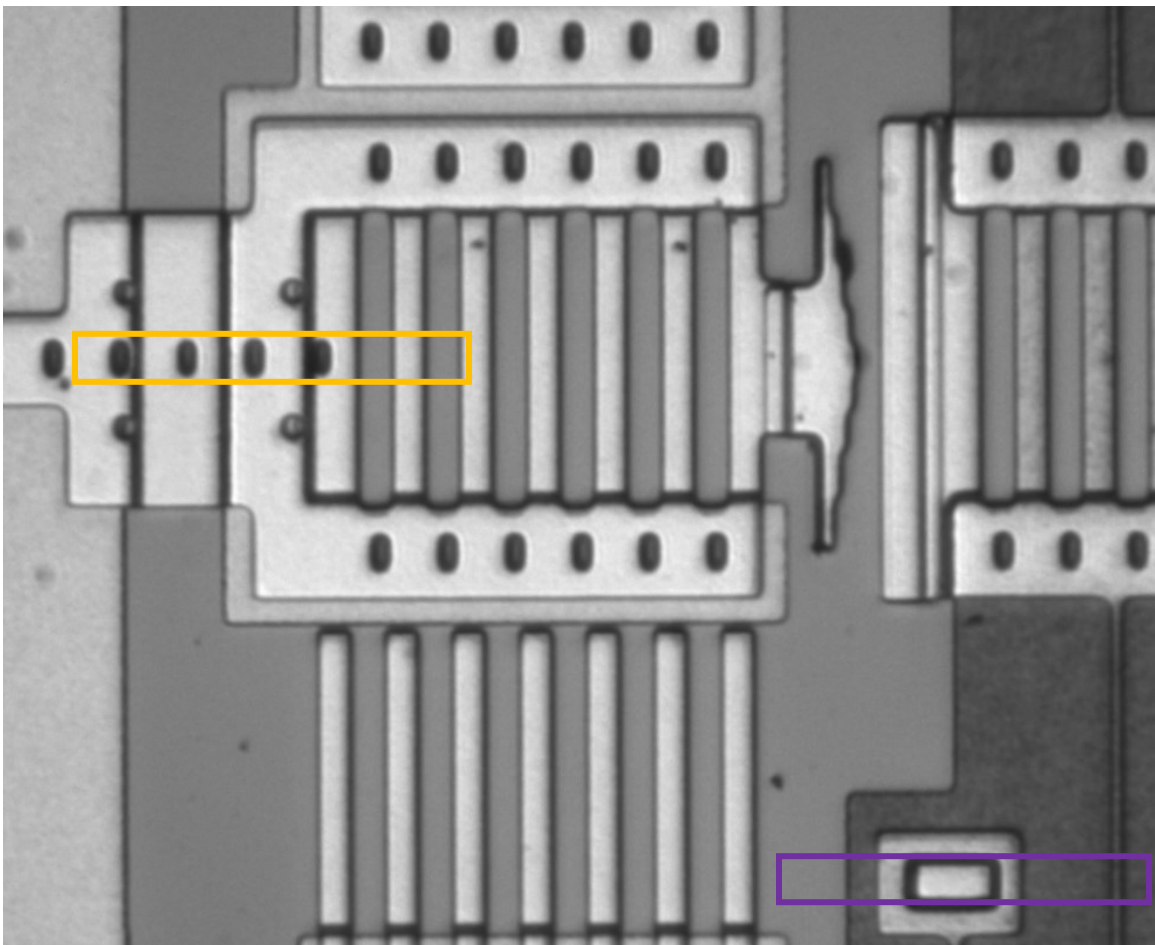


Figure 78: An irregular structure used in testing. The moving ROI (orange) and reference ROI (purple) are indicated.

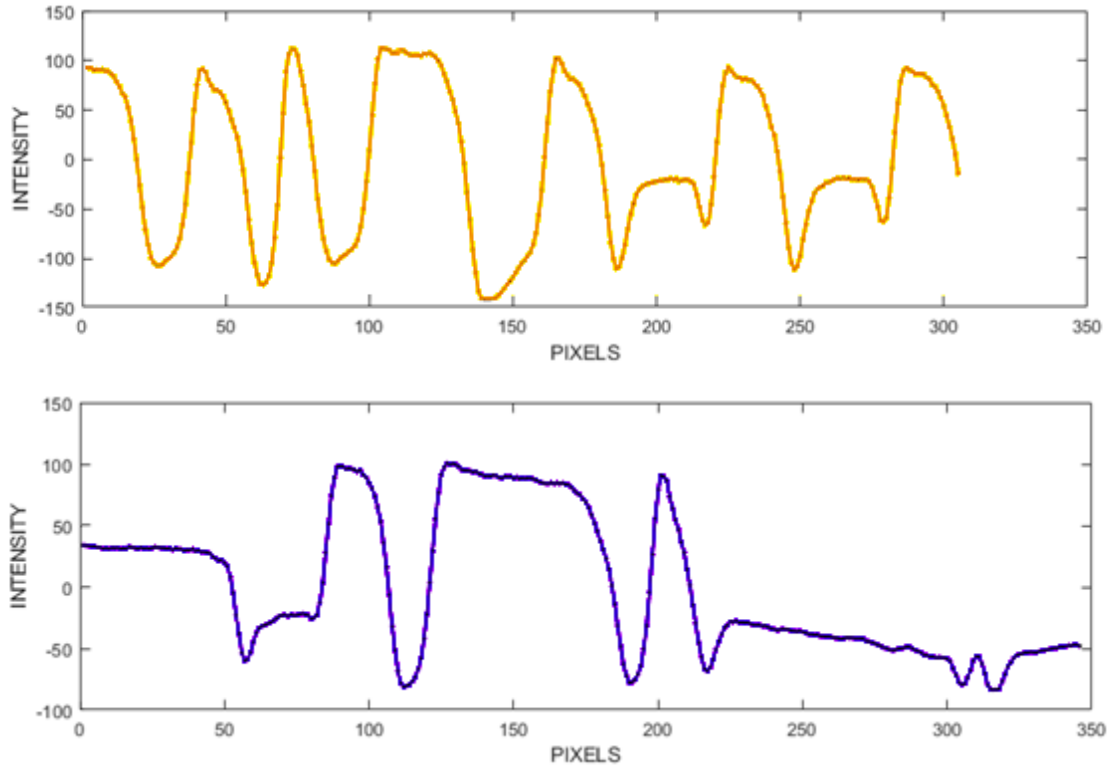


Figure 79: The irregular structure when converted to numeric values. Moving ROI above (in orange), reference ROI below (in purple)

Table 13: Measurement standard deviations for the XCorr and Yamahata for different ROIs, and the ratio.

ROI	XCorr Mean $\sigma$	Yamahata Mean $\sigma$	Yamahata/XCorr
Bars (40 px)	0.58	3.58	6.2
Dimples (40 px)	0.72	2.19	3.0
Non-Periodic (40 px)	0.83	13.4	16.1

The ROIs used in this test were 40 pixels high, similar in size to the test on the dimples completed in Section 6.3.4. In the case of the dimples, the XCorr performed about 3 times better, on average, than the Yamahata. When using the bars at this height, the XCorr performs about 6 times better than the Yamahata. In the non-periodic case, the XCorr performed about 16 times better, on average. This indicates that varied

structures tend to cause problems for the Yamahata, while not having any notable impact on the results from the XCorr. This further indicates that the XCorr has a large degree of versatility, as any vertically consistent structures that have a contrast in colour can be used as ROIs.

## 9. Conclusions & Future Work

The main objective of this thesis was the development of an algorithm with sub-nm displacement measurement precision for rectilinear motions which could be applied to MEMS devices. This system needed to have improved resolution, as well as improved versatility, when compared to the algorithm currently being used in the Dalhousie MEMS lab.

The newly developed algorithm is capable of resolutions of approximately  $1/700$  of a pixel, equal to a standard deviation of measurement of approximately 0.25 nm under ideal conditions. The algorithm was capable of clearly detecting steps of approximately 2.5 nm. However, as a step was considered detected if the average standard deviation of measurements was less than  $1/4$  of the step size, steps as small as 1 nm should be detectable using the new algorithm.

The algorithm uses cross-correlation of a greyscale image in order to evaluate displacements. User defined regions of interest are column averaged in order to reduce the image data to a 1-D array. 1-D simulations using an approximation of the intensity profiles found on MEMS devices, showed that it was possible to cross correlate signals to millipixel resolution through the use of a spline fit upsampling. The achieved resolution depends both on reducing the image noise and increasing the image SSSIG, a measure of the image contrast.

In addition, the simulated results showed that the signal needed to be pre-processed in order to produce millipixel measurements. It was found that the column average array of values needed to begin and conclude with zeros, and that having multiple leading and trailing zeros was beneficial. The algorithm first shifts the column average array so that the initial values are zeroed, and any slope between the leading and trailing values is removed from the signal.

In comparison with the previously used Yamahata Algorithm, the cross correlation algorithm performed better in all tests conducted. The new algorithm has been found to have, under ideal conditions, standard deviations of measurement 4.5 times smaller than the Yamahata algorithm used in the Dalhousie MEMS Lab, and under

non-ideal conditions the difference increases. Decreasing the number of periodic structures, decreasing the ROI height, using different structures including non-periodic structures revealed that the newly developed cross correlation algorithm is capable of handling other extreme circumstances better than the Yamahata algorithm previously used in the Dalhousie MEMS Lab.

The current algorithm has presented significant improvements under ideal and non-ideal conditions, however there are still a number of potential improvements that were not fully investigated in this thesis, and some of these may be investigated in the future.

Camera motions which occur over the course of an experiment cause slight shifts in the ROI. While the subtraction of the reference motion from the signal removes most of this noise, it may not fully do so. The current algorithm is able to detect displacements on the order of millipixels, and small effects such as this could create notable relative errors. It may be possible to modify the algorithm to better account for the motion of the image.

It was also shown that the colours of the image had a notable impact on the measured results. All images were taken with an RGB camera, and were then converted to greyscale. Using a camera designed for black and white imaging may produce more accurate result as there would be no Bayer filter and the camera SNR would be higher.

The camera used in this thesis operated in an 8-bit mode. While the camera used does have a 16-bit mode, it was not compatible with the LabVIEW data acquisition program. The use of a 16-bit images could increase the system performance.

Increasing the exposure time may also help to reduce the impact of noise, allowing for increased precision in the measurements. This was not investigated in this thesis as a result of incompatibilities with LabVIEW.



## Appendix A-1: XCorr MATLAB Program

```
close all
clearvars
clc
%-----
disp('Select image directory...')
disp(' ')

% Display a dialog box for the user to specify a directory.
path = uigetdir;
cd(path)

file_extension = 'png';

default_data_filename = 'xcorr_data';
data_file_name = input(['Enter name for the output data file ('
num2str(default_data_filename) ' is default) ... '], 's');
if isempty(data_file_name)
    data_file_name = default_data_filename;
end

default_pics_per_group = 10;
pics_per_group= input(['Enter num pics per groups ('
num2str(default_pics_per_group) ' is default) ... ']);
if isempty(pics_per_group)
    pics_per_group = default_pics_per_group;
end

default_sub_pix_res = .0001;
sub_pix_res = input(['Enter sub pixel resolution ('
num2str(default_sub_pix_res) ' is default) ... ']);
if isempty(sub_pix_res)
    sub_pix_res = default_sub_pix_res;
end

umPerPixel = 0.163*50/50;
limitation = umPerPixel*sub_pix_res;

%-----
--

% Creates list of all images in folder with desired filename and
extension
list_of_images = dir(strcat(path, '\*.', file_extension));
num_pics = size(list_of_images,1);
num_groups=num_pics/pics_per_group;
display(['# of groups = ', num2str(num_groups)]);

% Create an array of the file names of images.
file(1:num_pics,:) = char(list_of_images(1:num_pics).name);

% OPTIONAL Extract image_num from file name.
group_num = file(:,1);
```

```

% group_num = str2num(group_num);

% Read in first image.
first_image = imread(file(1,:));

% Convert uint8 to double, avoid 0-255 rounding errors
first_image=im2double(first_image);

% convert jpeg to grayscale.
if strcmp(file_extension,'jpg')== 1
    first_image = image_process(first_image,file_extension);
end

% imagestyle = 1; grayscale          <-- THESE ONES USUALLY
% imagestyle = 2; grayscale R only
% imagestyle = 3; grayscale G only  <-- THESE ONES USUALLY
% imagestyle = 4; grayscale B only
% imagestyle = 5; grayscale B+G only
% imagestyle = 6; grayscale R+G only

imagestyle = 1;

imagestyle_text ={'Grayscale','Red only','Green only','Blue only',
'Green + Blue', 'Green + Red'};
display(['Image Style = ',imagestyle_text{imagestyle}]);

first_image = image_process(first_image,file_extension,imagestyle);

%-----
--
%select ROI using imcrop (1) or use predefined ROIs (0)
choose_ROI_mov=0;
choose_ROI_ref=0;
%pre_def_roi_mov = [325,220,383,228];    %good mov
pre_def_roi_mov = [290,706,93,163];    %bad mov
%pre_def_roi_mov = [327,218,381,214];    %bad mov
pre_def_roi_ref= [375,608,494,233];    %good ref
%pre_def_roi_ref= [290,605,440,237];    %bad ref

if choose_ROI_mov
    disp('Select move ROI ...')
    figure('Color',[.8 .8 1])
    title('select MOVING comb crop box (double click to
finalize)','Color',[0 0 1],'FontSize',18),hold on
    % Disable the warning that appears when the imcrop function is used
    warning('off','images:initSize:adjustingMag');

    % Crop image by user input.
    [roi_mov, rect_mov]=imcrop(first_image);
    rect_mov = round(rect_mov);
else
    % OPTIONAL Crop images by predefined rectangles
    disp('Using predefined mov ROI ...')
    [roi_mov, rect_mov]=imcrop(first_image,pre_def_roi_mov);
end

```

```

if choose_ROI_ref
    disp('Select ref ROI ...')
    figure('Color',[1 0.8 0.8])
    title('select REFERENCE comb crop box (double click to
finalize)','Color',[0.8 0 0],'FontSize',18),hold on
    % Disable the warning that appears when the imcrop function is used
    warning('off','images:initSize:adjustingMag');

    % Crop image by user input.
    [roi_ref, rect_ref]=imcrop(first_image);
    rect_ref=round(rect_ref);
else
    % OPTIONAL Crop images by predefined rectangles
    disp('Using predefined ref ROI ...')
    [roi_ref, rect_ref]=imcrop(first_image,pre_def_roi_ref);
end

% close open figs
close all

fig_roi_mov = figure('Name','Moving
ROI','units','inches','Position',[5,5,6,6],'Color',[.8 .8 1]);
imshow(roi_mov);
title('MOVE crop box','Color',[0 0 0.8])

fig_roi_ref = figure('Name','Reference
ROI','units','inches','Position',[11,4,6,6],'Color',[1 .8 .8]);
imshow(roi_ref);
title('REF crop box','Color',[0.8 0 0]);
drawnow;

%-----

% for timing:
% tStart = tic;

disp(' ')
disp('CALCULATING MOVING PROFILES...')
[column_averages_mov] = column_average(file, num_pics,
rect_mov,imagestyle);

% for timing
% delta_t = toc(tStart);
% display(['delta T = ',num2str(delta_t,'%6.1f'),' sec']);

disp('CALCULATING REFERENCE PROFILES...')
disp(' ')
[column_averages_ref] = column_average(file, num_pics,
rect_ref,imagestyle);

% close open figs
% close all

%calculate SSSIG in first images
Int_Grad = gradient(column_averages_mov(1,:));

```

```

Sum_Int_Grad = Int_Grad.*Int_Grad;
SSSIG_mov = sum(Sum_Int_Grad);

Int_Grad = gradient(column_averages_ref(1,:));
Sum_Int_Grad = Int_Grad.*Int_Grad;
SSSIG_ref = sum(Sum_Int_Grad);
display(['SSSIG      mov,ref = ',num2str(SSSIG_mov,'%6.3f'),' ',
',num2str(SSSIG_ref,'%6.3f')]);

% calculate stdev in first num_groups images
stdev_row = std(column_averages_mov(1:num_groups,:));
noise_mov = mean(stdev_row);

stdev_row = std(column_averages_ref(1:num_groups,:));
noise_ref = mean(stdev_row);

display(['Noise      mov,ref = ',num2str(noise_mov,'%6.4f'),' ',
',num2str(noise_ref,'%6.4f')]);
noise_ratio = [noise_mov/sqrt(SSSIG_mov),
noise_ref/sqrt(SSSIG_ref)];
display(['Noise_ratio mov,ref = ',num2str(noise_ratio(1),'%6.4f'),
', ',num2str(noise_ratio(2),'%6.4f')]);
disp(' ')

xstart = 1;
xend = length(column_averages_mov(1,:));
xendref = length(column_averages_ref(1,:));

%create arrays of the desired data
x_mov = (xstart:xend);
x_ref = (1:xendref);

%-----
--

disp('COMPUTING MOVING DISPLACEMENTS...')
Correlation_pixels_mov = cross_correlation(num_pics, xstart,xend,
column_averages_mov, sub_pix_res, x_mov, 0);

disp('COMPUTING REFERENCE DISPLACEMENTS...')
Correlation_pixels_ref = cross_correlation(num_pics, xstart,xendref,
column_averages_ref, sub_pix_res, x_ref, 1);
disp(' ');

%Use the lag array to find the actual displacement
Correlation_um_mov = Correlation_pixels_mov*umPerPixel;
Correlation_um_ref = Correlation_pixels_ref*umPerPixel;

%Compare the moving and reference comb for total displacement
Correlation_um_NET = Correlation_um_mov - Correlation_um_ref;

if mean(Correlation_um_NET)<0
    Correlation_um_NET = Correlation_um_NET*-1;
    disp('Inverting values to get +ve numbers...')

```

```

        disp(' ')
    end

    stdev_um_NET = std(Correlation_um_NET);

    group_avgs = zeros(1,num_pics);
    group_stdevs = zeros(1,num_pics);

    j=1;
    for i = pics_per_group:pics_per_group:num_pics
        group_avgs(j) = mean(Correlation_um_NET(i-pics_per_group+1:i));
        group_stdevs(j) = std(Correlation_um_NET(i-pics_per_group+1:i));
        %disp(['group mean = ',num2str(group_avgs(j)*1000,'%6.2f'),' +/- ',
        ',num2str(group_stdevs(j)*1000,'%6.2f'),' nm']);
        j=j+1;
    end

    mean_stdev = mean(group_stdevs(1:num_groups))*1000;
    mean_value = mean(group_avgs(1:num_groups))*1000;

    hi_total=0;
    lo_total=0;
    hi_count=0;
    lo_count=0;

    for i = 1:num_groups
        if group_avgs(i)*1000> mean_value
            hi_total = hi_total+group_avgs(i)*1000;
            hi_count = hi_count+1;
        else
            lo_total = lo_total+group_avgs(i)*1000;
            lo_count = lo_count+1;
        end
    end

    hi_avg = hi_total/hi_count;
    lo_avg = lo_total/lo_count;
    delta_avg = hi_avg- lo_avg;
    disp(['HI avg = ',num2str(hi_avg,'%6.2f'),' nm']);
    disp(['LO avg = ',num2str(lo_avg,'%6.2f'),' nm']);
    disp(' ');
    disp(['Group STEP = ',num2str(delta_avg,'%6.2f'),' nm +/- ',
    num2str(mean_stdev,'%6.2f'),' nm']);
    disp(' ');

    total_range = abs(max(Correlation_um_NET)-
    min(Correlation_um_NET))*1000;
    disp(['Max range = ',num2str(total_range,'%6.1f'),' nm']);
    disp(' ');

    %Define scale for plot
    ymax = 1.5*max(Correlation_um_NET)*1000;
    ymin = 1.5*min(Correlation_um_NET)*1000;

    %Plot displacements on nanometer scale

```

```

results_fig =figure;
set(gcf, 'Name', 'Results', 'units', 'inches', 'position', [2,4,6,5])
plot(Correlation_um_NET*1000, ':o', 'MarkerEdgeColor', [0 0
1], 'MarkerFaceColor', [0.5 0.5 1], 'Color', [0 0 1])
ax = gca;
ax.GridColor = [.6 .6 .6];
grid on
set(gcf, 'color', 'w');
yticks(-4:1:4)
xlabel("IMAGE NUMBER"), ylabel("MEASURED DISPLACEMENT (NM)")
ylim([ (max(ymin, -4)), max(ymax,4) ])
title(['Output in nm, \sigma = ', num2str(mean_stdev, '%4.2f'), ...
' nm, (SSSIG = ', num2str(SSSIG_mov, '%6.3f'), ...
' , noise = ', num2str(noise_mov, '%6.4f'), ...
' , ratio = ', num2str(noise_ratio(1), '%6.4f'), ' )'])

%-----
show_groups=1;
if show_groups
group_x = 1:num_groups;
group_result=zeros(1,num_groups);
group_noise=zeros(1,num_groups);
for i = 1:num_groups
group_result(i)=group_avgs(i)*1000;
group_noise(i)=group_stdevs(i)*1000;
end

groups_fig =figure;
set(gcf, 'Name', 'Results
Grouped', 'units', 'inches', 'position', [8,4,6,5])

errorbar(group_x,group_result,group_noise, 'vertical', 'o', 'MarkerEdgeCol
or', 'b', 'MarkerFaceColor', [0.5 0.5 1], 'Color', [0 0 1])
ax = gca;
ax.GridColor = [.6 .6 .6];
grid on
yticks(-4:1:4)
xlim([0 9])
ylim([ (max(ymin, -4)), max(ymax,4) ])
title(['Output in nm, \sigma = ', num2str(mean_stdev, '%4.2f'), ...
' nm, (SSSIG = ', num2str(SSSIG_mov, '%6.3f'), ...
' , noise = ', num2str(noise_mov, '%6.4f'), ...
' , ratio = ', num2str(noise_ratio(1), '%6.4f'), ' )'])
end
%-----
savefigs = 0;
if savefigs
display('Saving to output_nm.png ...');
saveas(results_fig, '../Xcorr/output_nm.png');
end

% Select only images with correct file extenstion.
list_of_data_files = dir([(data_file_name), '*.csv',]);
num_data_files = size(list_of_data_files,1);
% Number of data files in folder.
disp(' ')

```

```

disp(['DATA FILE WILL BE SAVED AS
"',data_file_name,'_',num2str(num_data_files +1),'.csv" in folder
containing images'])
disp(' ')

% Header names.
list{1} = strcat('Move width',',',',num2str(round(rect_mov(3))),...
',',',',',',',',num2str(round(rect_ref(3))));
list{2} = strcat('# of images',',',',num2str(num_pics));
list{3} = strcat('# of groups',',',',num2str(num_groups));
list{4} = strcat('Group STEP = ',',',',num2str(delta_avg,'%6.2f'),',',',',
nm, +/- , ', num2str(mean_stdev,'%6.2f'),',',',', ' nm');
list{5} = strcat('SSSIG: mov &
ref',',',',num2str(SSSIG_mov,'%6.0f'),',',',num2str(SSSIG_ref,'%6.0f'));
list{6} = strcat('Noise: mov &
ref',',',',num2str(noise_mov,'%6.2f'),',',',num2str(noise_ref,'%6.2f'));
list{7} = strcat('Ratio: mov & ref',',',',num2str(noise_ratio(1),'
%6.4f'),',',',num2str(noise_ratio(2),'%6.4f'));
list{8} = strcat(' ');
list{9} = strcat('Group #',',',',', 'Mov nm',',',',', 'Ref nm',',',',', 'Image
#',',',',', 'Net nm',',',',', 'Grp Avg',',',',', 'Grp stdev');

% Print header names
print_file = strcat(data_file_name,'_',num2str(num_data_files +
1),'.csv');
fid = fopen(print_file,'w');
for i = 1:9
    list_print = list{i};
    fprintf(fid,'%s\r\n',list_print);
end
fclose(fid);

image_num = linspace(1,num_pics,num_pics);

% Create matrix of data to print
printing_data(:,1) = image_num;
printing_data(:,2) = Correlation_um_mov(1:end)*1000;
printing_data(:,3) = Correlation_um_ref(1:end)*1000;
printing_data(:,4) = image_num;
printing_data(:,5) = Correlation_um_NET(1:end)*1000;
printing_data(:,6) = group_avgs(1:end)*1000;
printing_data(:,7) = group_stdevs(1:end)*1000;
dlmwrite(print_file, printing_data,'precision','%7.3f','--append')

disp('DONE.')

```

## Appendix A-2: Image Processing MATLAB Function

```
function I_proc = image_process(varargin)

    % I_proc = improc(I,ext,imagestyle) processes the image I
    according to
    % the file extension and imagestyle and returns the processed
    image
    %
    % pgm: only does grayscale & imagestyle can be omitted
    % png & jpg:
    %     imagestyle = 1; grayscale          <-- THESE ONES USUALLY
    %     imagestyle = 2; grayscale R only
    %     imagestyle = 3; grayscale G only  <-- THESE ONES USUALLY
    %     imagestyle = 4; grayscale B only
    %     imagestyle = 5; grayscale B+G only
    %     imagestyle = 6; grayscale R+G only

    % Receive input values and check for correct syntax.
    I = varargin{1};

    if size(I,3) == 1
        I_proc = I;
    else
        if size(varargin,2) == 1;
            error('Not enough input arguments. See help image_process')
        end

        if ischar(varargin{2})
            ext = varargin{2};
        else
            error('In image_proc(I,ext) ext must be a string. See
            help image_process')
        end
        switch size(varargin,2)
            case 2
                if strcmp(ext,'png')
                    error('Not enough input arguments. See help
                    image_process')
                end
            case 3
                imagestyle = varargin{3};
                if isempty(imagestyle)
                    error(['In image_process(I,ext,imagestyle)
                    imagestyle must be a '...'
                    'number. See help image_process'])
                end
            otherwise
                error('Incorrect number of input arguments. See help
                image_process')
            end

            % Process image.
            switch ext
```



```

    case 'pgm'
        I_proc = I;
    % case 'jpg'
        %I_proc = rgb2gray(I);
    case {'png', 'jpg'}
        switch imagestyle
            case 1
                % grayscale
                I_proc = imadjust(rgb2gray(I));
            case 2
                % grayscale R ONLY
                %I(:, :, 1) = zeros(size(I(:, :, 1)));
                I(:, :, 2) = zeros(size(I(:, :, 1)));
                I(:, :, 3) = zeros(size(I(:, :, 1)));
                I_proc = imadjust(rgb2gray(I));
            case 3
                % grayscale G ONLY
                I(:, :, 1) = zeros(size(I(:, :, 1)));
                %I(:, :, 2) = zeros(size(I(:, :, 1)));
                I(:, :, 3) = zeros(size(I(:, :, 1)));
                I_proc = imadjust(rgb2gray(I));
            case 4
                % grayscale B ONLY
                I(:, :, 1) = zeros(size(I(:, :, 1)));
                I(:, :, 2) = zeros(size(I(:, :, 1)));
                %I(:, :, 3) = zeros(size(I(:, :, 1)));
                I_proc = imadjust(rgb2gray(I));
            case 5
                % grayscale B+G
                I(:, :, 1) = zeros(size(I(:, :, 1)));
                %I(:, :, 2) = zeros(size(I(:, :, 1)));
                %I(:, :, 3) = zeros(size(I(:, :, 1)));
                I_proc = imadjust(rgb2gray(I));
            case 6
                % grayscale R+G
                %I(:, :, 1) = zeros(size(I(:, :, 1)));
                %I(:, :, 2) = zeros(size(I(:, :, 1)));
                I(:, :, 3) = zeros(size(I(:, :, 1)));
                I_proc = imadjust(rgb2gray(I));
            otherwise
                error('Unrecognized imagestyle. See help
image_process')
        end
    end
    otherwise
        error('Unrecognized extension. See help image_process')
    end

    % experimental gaussian filter
    filt = 0;
    sigma = .5;
    if filt
        I_proc = imgaussfilt(I_proc, sigma);
    end

end

```

## Appendix A-3: Column Average MATLAB Function

```
function [crossSections] = column_average(file, num_pics,  
rect,imagestyle)  
  
%Initialize output variable  
crossSections = [];  
  
for i = 1:num_pics  
    % Read in the image.  
    ith_image = imread(file(i,:));  
  
    % Convert uint8 to double, avoid 0-255 rounding errors  
    ith_image=im2double(ith_image);  
  
    % Process the image.  
    ith_image = image_process(ith_image,'png',imagestyle);  
  
    %Crop image using previously defined rectangle.  
    roi = imcrop(ith_image,rect);  
  
    % Extract row average profile from cropped images.  
    cross_section = mean(roi);  
  
    crossSections = [crossSections;cross_section];  
end  
  
end
```

## Appendix A-4: Cross Correlation MATLAB Function

```
function [interpolatedLagArray] = cross_correlation(num_pics,
xstart,xend, column_averages, sub_pix_res, x, is_ref)

    y1 = column_averages(1,xstart:xend);

    y1_raw = y1;

    % calc end points for de-offset and de-trending
    yleft = mean(y1(xstart:xstart+2));
    yright = mean(y1(xend-2:xend));
    disp(['... y trend = ',num2str(yright-yleft,'%6.0f'),' /255']);

    % subtract offset
    y1 = y1-yleft;

    % de-trend
    for k = 1:xend-xstart+1
        y1(k) = y1(k) - k/(xend-xstart)*(yright-yleft);
        y1_trend(k) = yleft + k/(xend-xstart)*(yright-yleft);
    end

    y_flat = y1;

    interpolatedLagArray = [];

    for i = 1:num_pics
        y2 = column_averages(i,xstart:xend);

        y2_raw = y2;

        % calc end points for de-offset and de-trending
        yleft = mean(y2(xstart:xstart+2));
        yright = mean(y2(xend-2:xend));

        % subtract offset
        y2 = y2-yleft;

        % de-trend
        for k = 1:xend-xstart+1
            y2(k) = y2(k) - k/(xend-xstart)*(yright-yleft);
        end

        y2_flat = y2;

        %use xcorr to identify the shift to the nearest pixel
        max_num_pix_shift=100;
        [xcorrdata, lagarray] = xcorr(y1,y2,max_num_pix_shift);
        [~,xcorrpeak] = max(abs(xcorrdata));
        lag = lagarray(xcorrpeak);
```

```

%isolates data around the peak of xcorrdata for plotting
centered_array = (lag-4:lag+4);
xcorrdata_sample = xcorrdata(xcorrpeak-4:xcorrpeak+4);

%creates a spline fit of the data, in order to interpolate the
data
fine_array = (lag-4:sub_pix_res:lag+4);
splinefit = spline(centered_array,xcorrdata_sample,fine_array);

%finds the peak of the new spline
[~,finepeak] = max(abs(splinefit));
interpolatedLag = fine_array(finepeak);

%calculate total displacement
interpolatedLagArray = [interpolatedLagArray,interpolatedLag];

end

%figures showing the mean(roi), lag data, and a zoom in of the
spline

my_fig = figure('WindowState','normal');
set(gcf,"color","white")
if is_ref

set(gcf, 'Name', 'Reference', 'units', 'inches', 'position', [1,1,6,9])
else
    set(gcf, 'Name', 'Moving', 'units', 'inches', 'position', [10,1,6,9])
end

subplot(3,1,1)
plot(x,y1_raw,'r',x,y2_raw,'b',x,y1_trend,':m')
xlim([0,xend])
if is_ref
    title('REFERENCE Raw Data, 1st & last image')
else
    title('MOVING Raw Data, 1st & last image')
end

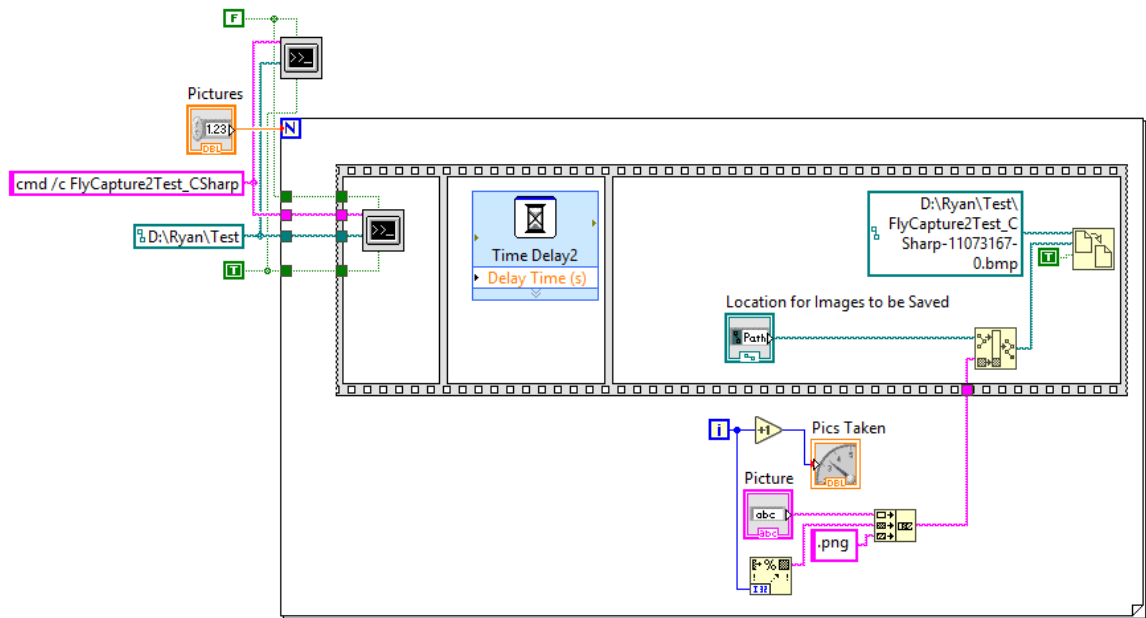
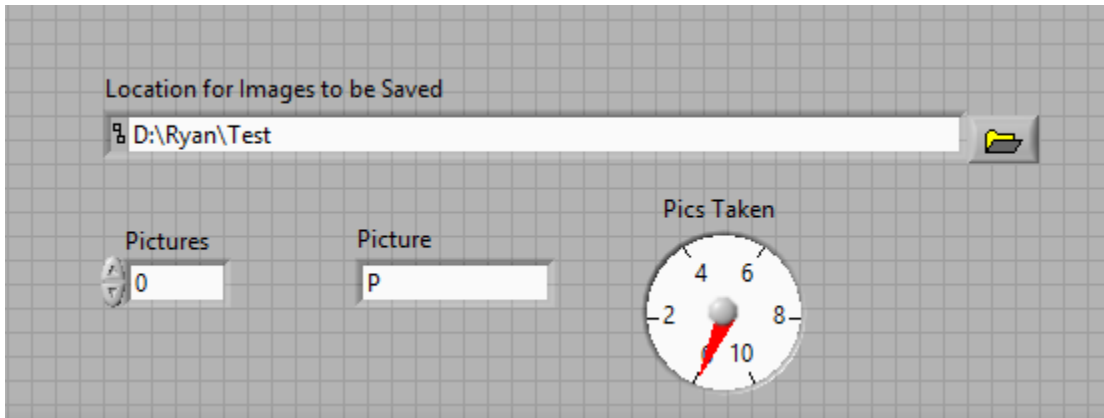
subplot(3,1,2)
plot(x,y_flat,'r',x,0*x,':k')
xlim([0,xend])
xlabel("PIXEL"),ylabel("INTENSITY")
if is_ref
    title('REFERENCE Corrected Data, 1st & last image')
else
    title('MOVING Corrected Data, 1st & last image')
end

subplot(3,1,3)
plot(lagarray,xcorrdata,'go')
% findpeaks(xcorrdata,'MinPeakProminence',200)
title('Cross correlation')

end

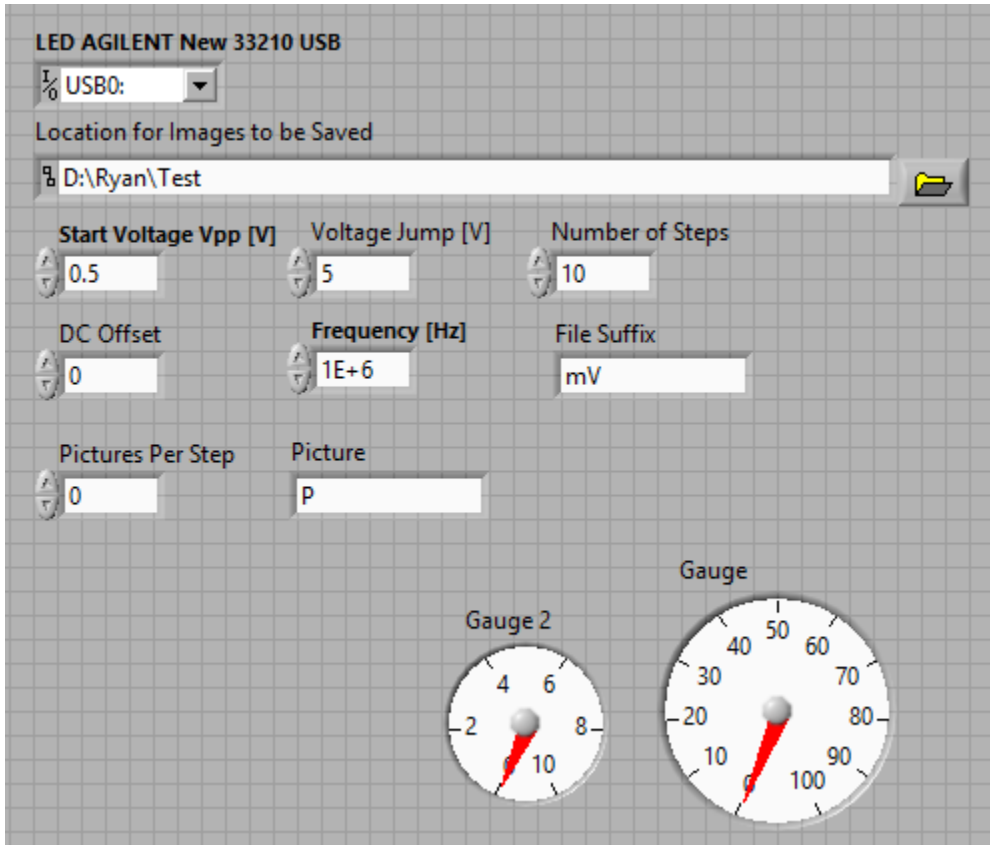
```

# Appendix B-1: Image Capture Only LabVIEW VI

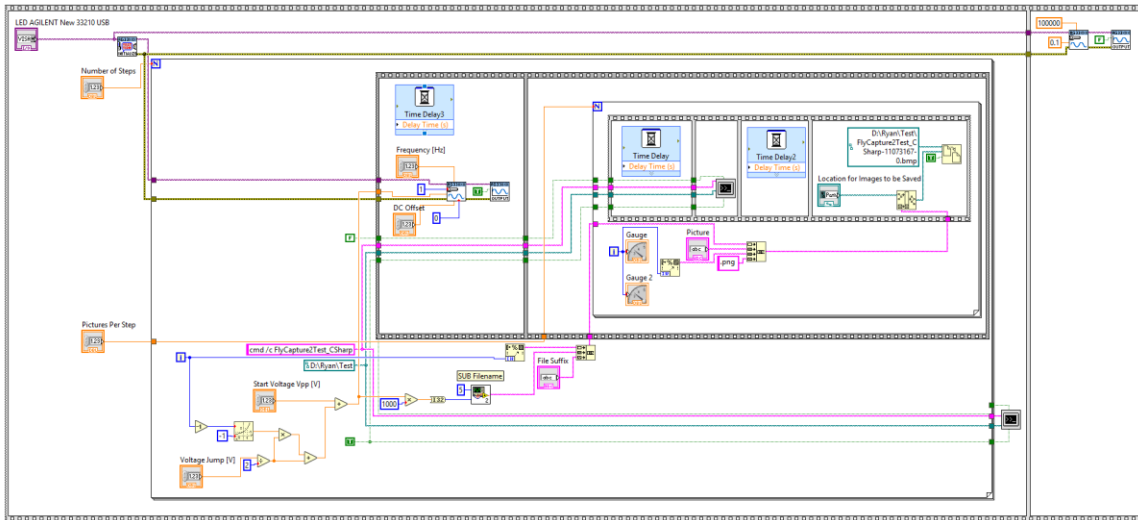


## Appendix B-2: Function Generator and Image Capture LabVIEW VI

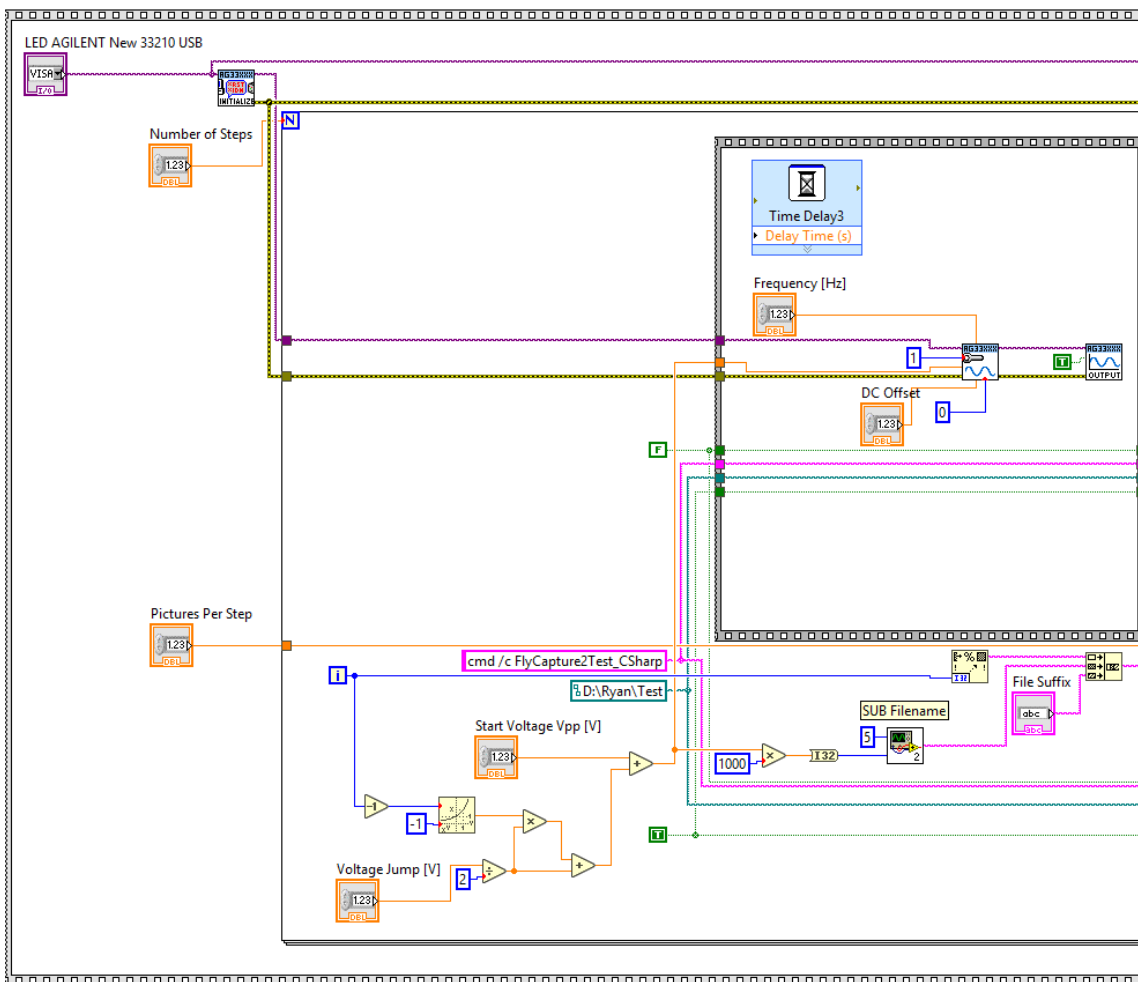
### Front Panel



## Block Diagram



## Block Diagram (Left)







## References

- [1] V. Lindroos, M. Tilli, A. Lehto and T. Motooka, *Handbook of silicon based MEMS*, 1st ed. Norwich, N.Y: William Andrew, 2010.
- [2] G. Kumar and A. Raman, "Pressure sensor based on MEMS nano-cantilever beam structure as a heterodielectric gate electrode of dopingless TFET", *Superlattices and Microstructures*, vol. 100, pp. 535-547, 2016. Available: 10.1016/j.spmi.2016.10.010.
- [3] C. Arthur, N. Ellerington, T. Hubbard and M. Kujath, "MEMS earthworm: a thermally actuated peristaltic linear micromotor", *Journal of Micromechanics and Microengineering*, vol. 21, no. 3, p. 035022, 2011. Available: 10.1088/0960-1317/21/3/035022.
- [4] Hoang, L. Wu, J. Golinval, M. Arnst and L. Noels, "Stochastic Multiscale Model of MEMS Stiction Accounting for High-Order Statistical Moments of Non-Gaussian Contacting Surfaces", *Journal of Microelectromechanical Systems*, vol. 27, no. 2, pp. 137-155, 2018. Available: 10.1109/jmems.2018.2797133.
- [5] N. Tambe and B. Bhushan, "Friction model for the velocity dependence of nanoscale friction", *Nanotechnology*, vol. 16, no. 10, pp. 2309-2324, 2005. Available: 10.1088/0957-4484/16/10/054.
- [6] B. Pan, "Digital image correlation for surface deformation measurement: historical developments, recent advances and future goals", *Measurement Science and Technology*, vol. 29, 2018. Available: 10.1088/1361-6501/aac55b
- [7] B. Pan, "An active imaging digital image correlation method for deformation measurement insensitive to ambient light", *Optics and Laser Technology*, vol. 44, no. 1, pp. 204-209, 2012. Available: 10.1016/j.optlastec.2011.06.019.
- [8] H. Bruck, S. McNeill, M. Sutton and W. Peters, "Digital image correlation using Newton-Raphson method of partial differential correction", *Experimental Mechanics*, vol. 29, issue 3, pp. 261-267, 1989. Available: 10.1007/BF02321405.
- [9] G. Stoilov, V. Kavardzhikov and D. Pashkouleva, "A Comparative Study of Random Patterns for Digital Image Correlation", *Journal of Theoretical and Applied*

- Mechanics*, vol. 42, no. 2, pp. 55-66, 2012. Available: 10.2478/v10254-012-0008-x.
- [10] G. Bomarito, J. Hochhalter, T. Ruggles and A. Cannon, "Increasing accuracy and precision of digital image correlation through pattern optimization", *Optics and Lasers in Engineering*, vol. 91, pp. 73-85, 2017. Available: 10.1016/j.optlaseng.2016.11.005.
- [11] P. Tu, "Digital image correlation with self-adaptive scheme for interpolation bias reduction", *Measurement Science and Technology*, vol. 28, no. 7, p. 075008, 2017. Available: 10.1088/1361-6501/aa70f8.
- [12] B. Pan et al., "Two-dimensional digital image correlation for inplane displacement and strain measurement: a review", *Measurement Science and Technology*, vol. 20, 2009. Available: 10.1088/0957-0233/20/6/062001.
- [13] S. Yagnamurthy, et al, "Mechanical and Ferroelectric Behavior of PZT-Based Thin Films", *Journal of Microelectromechanical Systems*, vol. 20, no. 6, pp. 1250-1258, 2011. Available: 10.1109/jmems.2011.2167666.
- [14] C. Robin, A. Vishnoi and K. Jonnalagadda, "Mechanical Behavior and Anisotropy of Spin-Coated SU-8 Thin Films for MEMS", *Journal of Microelectromechanical Systems*, vol. 23, no. 1, pp. 168-180, 2014. Available: 10.1109/jmems.2013.2264341.
- [15] T. Roland, S. Arscott, L. Sabatier, L. Buchaillet and E. Charkaluk, "Digital image correlation of metal nanofilms onSU-8 for flexible electronics and MEMS", *Journal of Micromechanics and Microengineering*, vol. 21, no. 12, p. 125005, 2011. Available: 10.1088/0960-1317/21/12/125005.
- [16] T. Berfield et al. "Micro- and Nanoscale Deformation Measurement of Surface and Internal Planes via Digital Image Correlation", *Experimental Mechanics*, vol. 47, pp. 51-62, 2007. Available: 10.1007/s11340-006-0531-2.
- [17] M. Naraghi, T. Ozkan, I. Chasiotis, S. Hazra and M. de Boer, "MEMS platform for on-chip nanomechanical experiments with strong and highly ductile nanofibers",

*Journal of Micromechanics and Microengineering*, vol. 12, no. 20, 2010.  
Available: 10.1088/0960-1317/20/12/125022.

- [18] A. Ya'akovitz, D. Copic, J. Beroz and A. Hart, "Nanoscale displacement measurement of microdevices via interpolation-based edge tracking of optical images", *Journal of Micromechanics and Microengineering*, vol. 23, no. 4, 2013.  
Available: 10.1088/0960-1317/23/4/045004.
- [19] H. Sugiura, S. Sakuma, M. Kaneko and F. Arai, "On-Chip Method to Measure Mechanical Characteristics of a Single Cell by Using Moiré Fringe", *Micromachines*, vol. 6, pp.660-673, 2015. Available: 10.3390/mi6060660.
- [20] D. Chang, S. Sakuma, K. Kera, N. Uozumi and F. Arai, "Measurement of the mechanical properties of single *Synechocystis* sp. strain PCC6803 cells in different osmotic concentrations using a robot-integrated microfluidic chip", *Lab on a Chip*, vol. 18, pp. 1241-1249, 2018. Available: 10.1039/c7lc01245d.
- [21] C. Yamahata, E. Sarajlic, G. Krijnen and M. Gijs, "Subnanometer Translation of Microelectromechanical Systems Measured by Discrete Fourier Analysis of CCD Images", *Journal of Micromechanics and Microengineering*, vol. 19, no. 5, pp. 1273-1275, 2010. Available: 10.1109/JMEMS.2010.2067445.
- [22]
- [23] J. Kokorian, F. Buja and W. van Spengen, "In-Plane Displacement Detection With Picometer Accuracy on a Conventional Microscope", *Journal of Microelectromechanical Systems*, vol. 24, no. 3, pp. 618-625, 2015. Available: 10.1109/jmems.2014.2335153.
- [24] J. Kokorian, F. Buja, U. Staufer and W. van Spengen, "An Optical In-plane Displacement Measurement Technique with Sub-Nanometer Accuracy Based on Curve-Fitting", *Journal of Microelectromechanical Systems*, vol. 23, no. 3, pp. 580-583, 2014. Available: 10.1109/MEMSYS.2014.6765707
- [25] J. Carter, A. Cowen, B. Hardy, R. Mahadevan, M. Stonefield and S. Wilcenski, *PolyMUMPs Design Handbook*, 11th ed. 2005.

- [26] L. Maiolo et al., "Flexible sensing systems based on polysilicon thin film transistors technology", *Sensors and Actuators B: Chemical*, vol. 179, pp. 114-124, 2013. Available: 10.1016/j.snb.2012.10.093.
- [27] K. Kant, A. Shukla, A. Sharma and P. Biwole, "Thermal response of poly-crystalline silicon photovoltaic panels: Numerical simulation and experimental study", *Solar Energy*, vol. 134, pp. 147-155, 2016. Available: 10.1016/j.solener.2016.05.002.
- [28] R. Hickey, M. Kujath, T. Hubbard, "Heat transfer analysis and optimization of two-beam microelectromechanical thermal actuators", *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, vol. 20, no. 3, pp. 971-974, 2002. Available: 10.1116/1.1468654.
- [29] N. Ellerington, B. Bschaden, T. Hubbard and M. Kujath, "Fourier analysis of blurred images for the measurement of the in-plane dynamics of MEMS", *Journal of Micromechanics and Microengineering*, vol. 22, no. 3, p. 035019, 2012. Available: 10.1088/0960-1317/22/3/035019.
- [30] S. Warnat, C. Forbrigger, T. Hubbard, A. Bertuch and G. Sundaram, "Thermal MEMS actuator operation in aqueous media/seawater: Performance enhancement through atomic layer deposition post processing of PolyMUMPs devices", *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, vol. 33, no. 1, pp. 01A126, 2015. Available: 10.1116/1.4902081.
- [31] B. Pan, H. Xie and Z. Wang, "Equivalence of digital image correlation criteria for pattern matching", *Applied Optics*, vol. 49, no. 28, p. 5501, 2010. Available: 10.1364/ao.49.005501.
- [32] P. Bing, X. Hui-min, X. Bo-qin and D. Fu-long, "Performance of sub-pixel registration algorithms in digital image correlation", *Measurement Science and Technology*, vol. 17, no. 6, pp. 1615-1621, 2006. Available: 10.1088/0957-0233/17/6/045.
- [33] P. Mazzoleni, F. Matta, E. Zappa, M. Sutton and A. Cigada, "Gaussian pre-filtering for uncertainty minimization in digital image correlation using numerically designed speckle patterns", *Optics and Lasers in Engineering*, vol. 66, pp. 19-33, 2015. Available: 10.1016/j.optlaseng.2014.08.004.

- [34] S. Wang, B. Guan, G. Wang and Q. Li, "Measurement of sinusoidal vibration from motion blurred images", *Pattern Recognition Letters*, vol. 28, no. 9, pp. 1029-1040, 2007. Available: 10.1016/j.patrec.2006.12.019.
- [35] B. Pan, "Bias error reduction of digital image correlation using Gaussian pre-filtering", *Optics and Lasers in Engineering*, vol. 51, no. 10, pp. 1161-1167, 2013. Available: 10.1016/j.optlaseng.2013.04.009.
- [36] Q. Ma and S. Ma, "Experimental investigation of the systematic error on photomechanic methods induced by camera self-heating", *Optics Express*, vol. 21, no. 6, p. 7686, 2013. Available: 10.1364/oe.21.007686.
- [37] B. Pan, W. Shi and G. Lubineau, "Effect of camera temperature variations on stereo-digital image correlation measurements", *Applied Optics*, vol. 54, no. 34, p. 10089, 2015. Available: 10.1364/ao.54.010089.
- [38] B. Bazarani, S. Warnat, A. Fine and T. Hubbard, MEMS squeezer for the measurement of single cell rupture force, stiffness change, and hysteresis", *Journal of Micromechanics and Microengineering*, vol. 27, no. 2, pp. 50-58, Available: 10.1088/1361-6439/27/2/025002.