

Project Number: MQP CF – MM09

MEMS for real-time imaging

A Major Qualifying Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

in Mechanical Engineering

by

Marc Balboa

Ivo Dobrev

Ryan Fossett

April 30th, 2009

Approved:

Prof. Cosme Furlong-Vasquez, Major Advisor

Table of Contents

Table of Figures	iii
List of Tables	vi
Table of Equations	vii
Acknowledgements.....	viii
Abstract.....	ix
1. Introduction	- 2 -
2. Background.....	- 5 -
2.1. MicroElectroMechanical Systems (MEMS).....	- 5 -
2.1.1. Bulk Micromachining and Etching.....	- 5 -
2.1.2. Surface Micromachining.....	- 6 -
3. Generation of Grayscale Pattern Variation using Interferometry.....	- 8 -
3.1. Interferometry.....	- 8 -
3.1.1. Single Exposure Mode.....	- 11 -
3.1.2. Double Exposure Mode	- 12 -
3.1.3. Resolution and Accuracy	- 13 -
3.2. MEMs Micro Mirror Array	- 17 -
4. Realization of pattern generation using MEMS	- 20 -
4.1. Simple Bimaterial Cantilever Beam.....	- 20 -
4.1.1. Analytical Model	- 20 -

4.1.1.1.	Theory, Parameters, and Calculations	- 20 -
4.1.1.2.	Uncertainty Analysis	- 21 -
4.1.2.	Computational Model	- 22 -
4.1.2.1.	Choosing an Element Type.....	- 22 -
4.1.2.2.	Determining the Element Size.....	- 23 -
4.1.2.3.	Results and Validation.....	- 24 -
4.2.	MEMS Device Bimaterial Beam Array	- 26 -
4.2.1.	MEMS Device Fabrication Process	- 26 -
4.2.2.	Computational Model	- 28 -
4.2.2.1.	Beam Response.....	- 29 -
4.3.	Experimental Results.....	- 31 -
5.	Conclusions and Recommendations	- 41 -
5.1.	Parametric Investigation and Recommendations	- 41 -
5.2.	Conclusions and future work	- 44 -
6.	References	- 46 -
	Appendix A: Derivation of Timoshenko's Analytical Model	- 48 -
	Appendix B: Analytical Calculations of a Bimaterial	- 50 -
	Appendix C: Matlab programs.....	- 53 -
	Appendix D: Micro Mirrors Experimental Setup	- 83 -
	Appendix E: Commercial Components Used in Thermal Loading Station.....	- 88 -

Table of Figures

Fig. 1. Thermal image of a human.....	- 2 -
Fig. 2. Critical Imaging IR camera.	- 2 -
Fig. 3. Linnik laser interferometer.	- 3 -
Fig. 4. Superposition principle illustration.	- 9 -
Fig. 5. Linnik interferometric setup.	- 10 -
Fig. 6. Linnik interferometric lab setup.	- 10 -
Fig. 7. Four step algorithm raw data.	- 12 -
Fig. 8. Iterations of 5-step algorithm. The resulting voltages are indicated.	- 15 -
Fig. 9. Distributions of surface measurements of a flat plate.	- 16 -
Fig. 10. MEMS generation of the letter "J", viewed using interferometry.	- 18 -
Fig. 11. Small distortions in the surface of the micro mirrors.	- 19 -
Fig. 12. Graphical representation of Timoshenko's solution.	- 21 -
Fig. 13. Meshed cantilever beam showing end constraints.	- 23 -
Fig. 14. Isometric view of FEM solution for simple cantilever.....	- 24 -
Fig. 15. Analytical and computational model comparison.	- 25 -
Fig. 16. Fabrication process used for bimaterial micro cantilever beams array.	- 27 -
Fig. 17. 50x magnified picture of bimaterial cantilever beam array.....	- 28 -
Fig. 18. Graphic representation of bimaterial cantilever beam from FEM model.....	- 28 -
Fig. 19. Meshed bimaterial pixel showing end constraints.....	- 29 -
Fig. 20. Isometric view of bimaterial pixel.....	- 29 -

Fig. 21. Beam deflection vs. temperature with linear best fit.....	- 30 -
Fig. 22. CAD model of thermal loading station.....	- 32 -
Fig. 23. Design realization of thermal loading station.....	- 32 -
Fig. 24. Step 1 of masking algorithm.....	- 33 -
Fig. 25. Step 2 of masking algorithm.....	- 34 -
Fig. 26. Step 3 of masking algorithm.....	- 35 -
Fig. 27. Step 4 of masking algorithm.....	- 36 -
Fig. 28. Step 5 of masking algorithm.....	- 36 -
Fig. 29. Raw masked data.....	- 37 -
Fig. 30. Phase unwrapped data collected using the interferometer.....	- 37 -
Fig. 31. A 3D view of the surface of a single beam.....	- 37 -
Fig. 32. Raw data at increasing temperature and corresponding experimental deflections.....	- 38 -
Fig. 33. Beam surfaces at varying temperatures.....	- 39 -
Fig. 34. Beam sensitivity comparison for FEM and experimental results.....	- 39 -
Fig. 35: Zoomed-in view of Fig. 34.....	- 40 -
Fig. 36. Deflection vs. thickness ratio from analytical solution.....	- 42 -
Fig. 37. Deflection vs. thickness ratio from FEM model.....	- 43 -
Fig. 38. Elastic modulus ratio vs. deflection.....	- 44 -
Figure 39: The micro mirror driving hardware.....	- 84 -
Figure 40: Positioning the micro mirror array.....	- 86 -
Figure 41: Micro mirror interface software.....	- 87 -

List of Tables

Table 1. Beam parameters.....	- 20 -
Table 2. Percent error contribution.	- 22 -
Table 3. Determination of mesh size.	- 24 -
Table 4. Beam deflection vs. temperature differential.....	- 30 -
Table 5. Comparison of experimental results with computational solution.	- 40 -
Table 6. Comparison of thermal resolutions.....	- 41 -
Table 7: Micro mirror experiment hardware specifications	- 83 -

Table of Equations

Equation 1	- 11 -
Equation 2	- 11 -
Equation 3	- 12 -
Equation 4	- 12 -
Equation 5	- 13 -
Equation 6	- 20 -

Acknowledgements

We would like to thank Professor Cosme Furlong for advising this project. We would like to acknowledge Insight Technologies, Inc. for their funding and feedback, and for allowing WPI to be a part of their cutting edge research and development. We also offer gratitude to Boston Micromachines Corporation for the micro mirrors and related equipment and software and for their cooperation. We extend a special thanks to Adriana Hera for her late hours in the design studio and her on-the-spot guidance to FEM modeling, and to Ellery Harington for providing us with the software used to acquire information from the interferometer. We also apologize to him for sending him on some programming 'wild goose chases', when our suggestions for changing his programs proved to be less than useful.

Abstract

This project investigates an innovative approach to imaging with Micro Electro-Mechanical Systems (MEMS) based devices. By using a Linnik interferometer and advanced phase unwrapping algorithms for processing data, the feasibility of generating high-resolution grayscale images in real-time was proven with an array of individually addressable MEMS micro-mirrors. Further investigations on a thermal imaging detector consisting of an array of pixels defined by surface micromachined bi-material beam structures were carried out. A thermal loading fixture was manufactured and incorporated into the interferometer setup, which was also optimized to provide high measuring resolution. Interferometric images were collected at several temperatures in order to determine the beams' response as a function of temperature, which successfully demonstrated the suitability of the detector to imaging with high-sensitivity and with a linear response. Experimental results were used with analytical and computational models to further predict the thermo-mechanical characteristics of the beams and to perform parametric investigations and optimization of their design. Further developments will consist of integrating the detector into a highly advanced, completely mechanical, imaging device having mK thermal resolution. The availability of such device will greatly improve current thermal imaging technology.

1. Introduction

This project investigates an innovative approach to thermal imaging using MEMS devices with laser interferometry, a technique that allows for the generation of continuous grayscale variations. The current state of the art thermal imaging devices, like the one shown on Fig. 2. Critical Imaging IR camera use CMOS and CCD sensors - digital devices which are only able to identify quantized levels of infrared radiation. An example of an image of a human infrared radiation signature read by such a sensor is shown in Fig. 1. Since this radiation is emitted from any warm body, these devices are subject to high levels of noise; even from the devices themselves. As a result the device requires expensive cooling systems in order to maintain high thermal resolutions, thus limiting them to stationary use. These cameras have digital resolution on the order of 25-30mK, which is insufficient for some of today's thermal imaging applications such as development of advanced circuitry and long range infrared detection devices for homeland security.

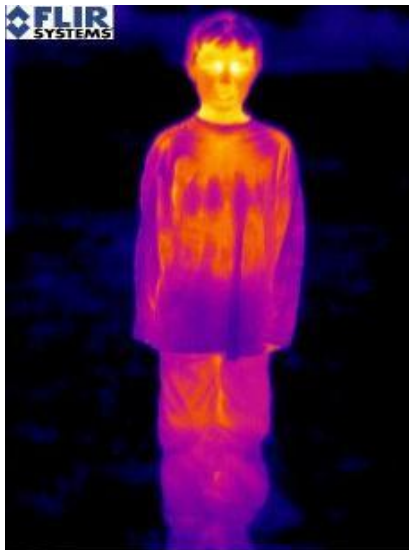


Fig. 1. Thermal image of a human.



Fig. 2. Critical Imaging IR camera.

Higher resolutions can be obtained using MEMS devices that react to infrared radiation. These devices use micro scale, bimaterial cantilever beams, which react to changes in temperature by deflecting out of plane. Naturally, the scale of these variations is on the order of nanometers, so highly sensitive measurement techniques with high resolution must be exploited.

Interferometry is a commonly used technique for measuring changes in position with sub-nanometer resolution. This project utilized the Linnik interferometer setup. Changes in optical path length of the object beam (shown in Fig. 3) create constructive and destructive interference resulting in continuous grayscale variations proportional to the position of the sample. Refer to the “Interferometry” section on page - 8 - of this paper for more information on how an interferometer works.

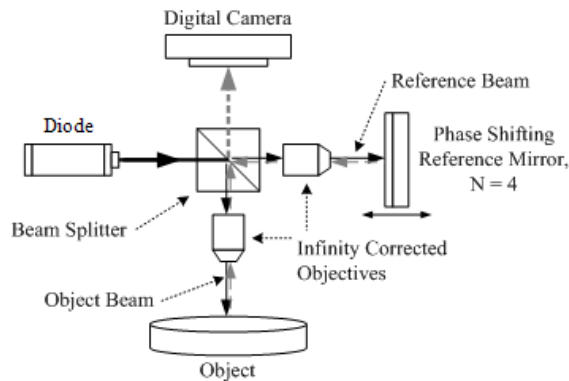


Fig. 3. Linnik laser interferometer.

The theory is that by using the bimaterial MEMS devices described above, deflections of the beams can generate thermally actuated grayscale patterns. However, before using this device, the ability of the interferometer to detect such pattern generation must be proven. This will be done using a commercially available micro mirror array. These mirrors are electronically actuated and can be driven out of plane to generate pixel-like patterns. Only after this successful

proof of concept will the response of thermally actuated pixels be tested, characterized, and analyzed.

2. Background

2.1. MicroElectroMechanical Systems (MEMS)

Advances in manufacturing processes have brought MicroElectroMechanical systems (MEMS) into the technological forefront and they are becoming increasingly popular for making precise mechanical measurements in engineering. Characterizing these devices, however, is a difficult task due to their microscopic scale. Fortunately, interferometry allows precise measurement of these devices allowing engineers to accurately analyze and improve them and to improve the processes by which they are made. A description of two of those processes follows (Hsu, 2002).

2.1.1. Bulk Micromachining and Etching

Bulk micromachining is a process by which material is removed from silicon wafer substrates in order to generate desired geometries. The primary method by which material is removed is a process called etching, where certain chemicals, or “etchants”, are exposed to the silicon wafers. The areas where these etchants contact the silicon substrate can be controlled by using a protective mask. The etching process can be broken down into two types: isotropic and anisotropic. Isotropic etching is independent of direction, and the etchant will eat away at the substrate uniformly in all directions. Naturally, this creates a less desirable and controllable geometry. Conversely, anisotropic etching is orientation-dependant and for this reason it is a more desirable method. Due to the crystalline structure of the substrate, it may be more resistant to chemical etching in some directions, explaining the orientation-dependency. Utilization of this principle is what allows anisotropic etching to generate more predictable geometry. Naturally, this orientation-dependent corrosion resistance results in different etching rates.

There are three main disadvantages to anisotropic etching. The rates are slower than that of isotropic etching, the etching rates increase with temperature, and elevated temperatures have limiting effects on the masking materials.

An important aspect in etching is the *Selectivity Ratio*. This is defined as the ratio of the rate of etching of silicon to another material, for a given etchant. For instance, Silicon Dioxide has a selectivity ratio of 1000 (meaning that it etches at a rate slower than silicon by a factor of 1000) while Silicon Nitride has a ratio of 10,000, for KOH. These selectivity ratios are important in determining which substances to use for masking. However, it is important to note that while masks etch at slower speeds than substrates they will still be subject to corrosion if left in the etchant for too long; timing is critical.

2.1.2. Surface Micromachining

Surface micromachining is a process by which microstructures are formed using layer-by-layer deposition. Generally, a main structure is created by depositing layers of silicon, silicon nitrate, or in most cases, polysilicon. These layers can range in thickness from about 2 μm to 5 μm . At the same time, sacrificial layers are added. These sacrificial layers are used for structural purposes, but are later removed, usually by etching. They are an integral part of the surface micromachining process that, once removed, can free structures to move, such as in the case of micro cantilever beams. A common material used for these layers is silicon dioxide due to its significantly faster etch rate than that of the aforementioned silicon compounds.

Such as in the case of bimaterial strips, additional compounds can be deposited atop the silicon by means of chemical vapor deposition (or several other processes) and these layers can range in size from 0.1 μm to 5 μm thick. Naturally, such thin films are subject to mechanical

problems. The two main failure modes are peeling due to improper adhesion of layers and shear failure. While the former is self-explanatory, the latter failure mode stems from the adhesion of layers with differing coefficients of thermal expansion (CTE). While the difference in CTEs is the main principle behind bimaterial strips, the difference in expansion between the two layers generates high stress at the shared boundary and the result is often shear failure as the two layers slide past one another.

3. Generation of Grayscale Pattern Variation using Interferometry

3.1. Interferometry

Interferometry deals with extracting information based on the resulting patterns created by the interference of two or more waves. The basic principle behind interferometry is the one of superposition, which states that the net response caused by several stimuli is the sum of the responses which would have been caused by each individual stimulus. One specific application of interferometry is the extraction of information about the difference between the path lengths of two waves based on the intensity of the wave that results from their interference. This is based on the fact that when two waves with the same frequency are combined, the resulting pattern will depend on their phase difference. In other words, waves (red and purple lines on Fig. 4) with small or no phase shift will create a constructive interference (red line on the top of Fig. 4) resulting in greater intensity wave at the point of interference. On the other hand, 180deg of phase shift will result in destructive interference, thus zero intensity wave at the point of interference.

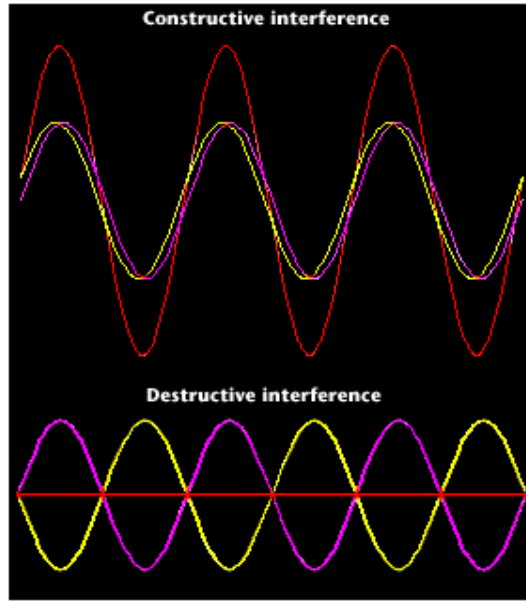


Fig. 4. Superposition principle illustration.

To reverse the process, the intensity at the point of interference can be read and the information about the phase shift of the two waves can be extracted. Knowing the wavelength of the waves, the difference in the wave paths can be calculated. This allows for comparison of wave paths with a great amount of resolution. If the length of one of the wave paths is known then, based on the difference extracted from the interferometric analysis of the two waves, the length of the other wave path can be calculated.

A diagram of an experimental setup that allows for such precise wave path length measurement is shown in Fig. 5. A picture of the actual interferometric setup is shown on Fig. 6.

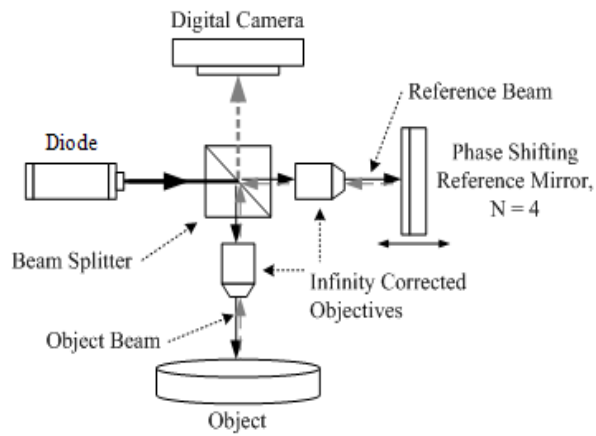


Fig. 5. Linnik interferometric setup.

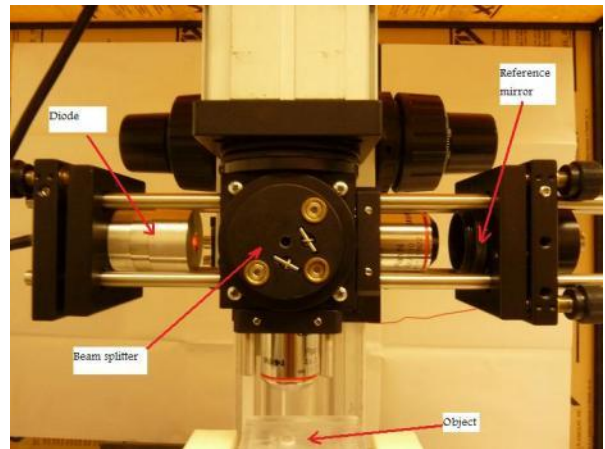


Fig. 6. Linnik interferometric lab setup.

The principle of operation of the Linnik interferometric setup is as follows:

1. Diode emits a beam directed at the beam splitter.
2. The beam is split into two beams – the object beam and the reference beam.
3. The object beam is deflected towards the object, reflects off the sample in the direction of the camera.
4. The reference beam continues towards the reference mirror, reflects off the mirror in the direction of the beam splitter. At the beam splitter the beam is redirected towards the camera.
5. At the camera a interferometry pattern is created and is read by the camera's sensor as changes in the intensity across the pixels of the sensor.
6. This results in continuous grayscale variation corresponding in the intensity of the wave that is the product of the destructive or constructive interference pattern due to the incoming waves at each pixel.

3.1.1. Single Exposure Mode

The theory of interference predicts that the intensity of the wave at every point of a surface, on which interference pattern exists, will be governed by the following formula:

$$I_n(x, y) = I_o(x, y) + I_r(x, y) + 2\sqrt{I_o(x, y)I_r(x, y)} \cos[\alpha(x, y)] , \quad \text{Equation 1}$$

Where I_o is the object beam intensity, I_r is the reference beam intensity, and $\alpha(x,y)$ is the phase difference between the two waves at a particular position on the plane of the sensor. However this equation consists of three unknowns, one of which is the desired parameter $\alpha(x,y)$. To solve for this parameter, at least three sets of data needs to be acquired allowing for the generation of a set of three equations. However there are algorithms that use more sets of equations so that they have greater accuracy and are less sensitive to misalignments in the physical setup. In our lab setup we apply a 4-step algorithm to determine the position of the object. The data for the set of equations is generated by moving the reference mirror at 4 known positions along the direction of transmission of the waves. Applying this algorithm for the specific configuration for the Linnik interferometer setup we have:

$$I_n(x, y) = I_o(x, y) + I_r(x, y) + 2\sqrt{I_o(x, y)I_r(x, y)} \cos[\Omega(x, y) + \theta_n] , \quad \text{Equation 2}$$

Where, $\Omega(x,y)$ is the fringe-locus function in the (x,y) space, and θ_n is the n -th phase-step. $\Omega(x,y)$, related to the object's shape and deflections, $L_z(x,y)$, is obtained, for the case of 4-step algorithm ($n = 4$) as follows:

$$\Omega(x, y) = \tan^{-1} \left[\frac{I(x, y)_4 - I(x, y)_2}{I(x, y)_1 - I(x, y)_3} \right], \quad \text{Equation 3}$$

Based on that the put of plane shape and deflection of the object's surface is defined as follows:

$$L_z(x, y) = \frac{\lambda}{4\pi} \cdot \Omega(x, y). \quad \text{Equation 4}$$

Where, λ is the wavelength of the waves.

Fig. 7 shows a set of 4 images corresponding to the 4 steps used in the algorithm:

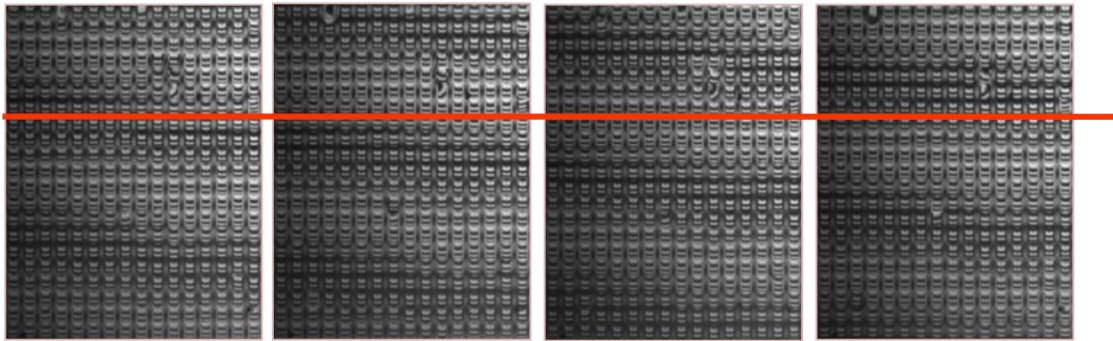


Fig. 7. Four step algorithm raw data.

The red line is shown for reference for the position of the interferometry pattern fringes at each position of the reference mirror. The position of the fringes changes according to the position of the reference mirror at every frame.

3.1.2. Double Exposure Mode

While this algorithm is useful for absolute measurements of the shape and the deflection of the surface of the object it is further modified for the specific application of the interferometric setup

by referencing all frames $(I(x, y)_{1-4})$ to an initial frame acting as a datum for the all the consecutive grayscale intensity distributions read by the camera. This means that using a referencing technique like this we can measure only changes in displacement rather than absolute positions. This allows for reading of very small displacements with respect to the initial shape of the surface of the object.

3.1.3. Resolution and Accuracy

A crucial part for the accuracy of the system is the accuracy of the position of the reference mirror at every single step of the algorithm. Since the mirror is driven by a piezo-based actuator we used a specialized 5-step algorithm. The algorithm is based on minimizing the following value:

$$\cos[\alpha(x, y)] = \frac{I(x, y)_1 - I(x, y)_5}{2(I(x, y)_2 - I(x, y)_4)}, \quad \text{Equation 5}$$

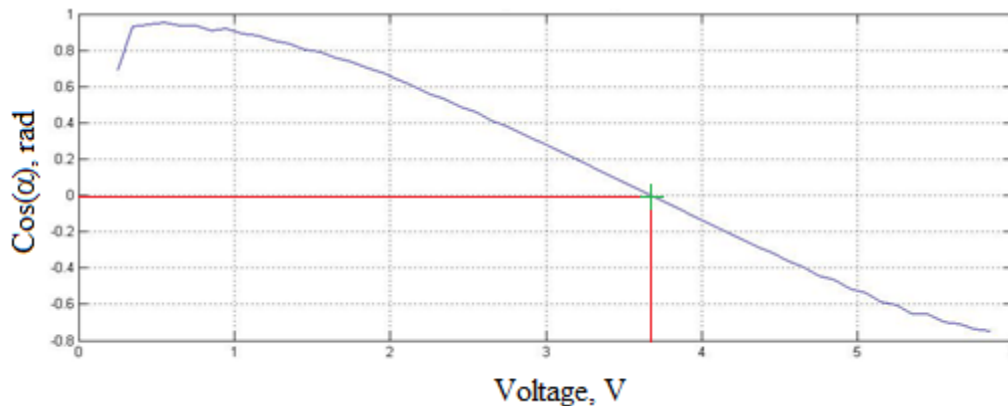
The procedure for using the above formula for calibrations of the piezo is as follows:

1. The 5th frame is generated by sending specified voltage V_i to the piezo.
2. The rest of the frames from 4-1 are generated by a corresponding voltage $\frac{3}{4}V_i, \frac{1}{2}V_i, \frac{1}{4}V_i$ and $0V_i$. Frame 3 is excluded from the equation and is used for error checking of the rest of the algorithm.
3. The value of the parameter $\cos(\alpha(x, y))$ for the specific voltage V_i is calculated.

- Steps from 1-3 is repeated for a range voltages and the resulting values are plotted against the corresponding voltages.
- The cross-section of the resulting curve with the x-axis (the voltage) is found along with a narrower range of voltages.
- The process is iterated until the step between the voltages is reduced below the resolution of the DAC system that is used to drive the piezo-actuator via the PC.

A detailed description of the flow diagram and code for this algorithm are shown in Appendix C:

. Fig. 8 shows three iterations of the described algorithm. The x-axis represents the voltage of the piezo-actuator in volts and the y-axis represents the value of $\cos(\alpha(x, y))$ at the measured location. The zero of each curve is indicated by a green cross-mask and its x and y coordinates are indicated by the red lines.



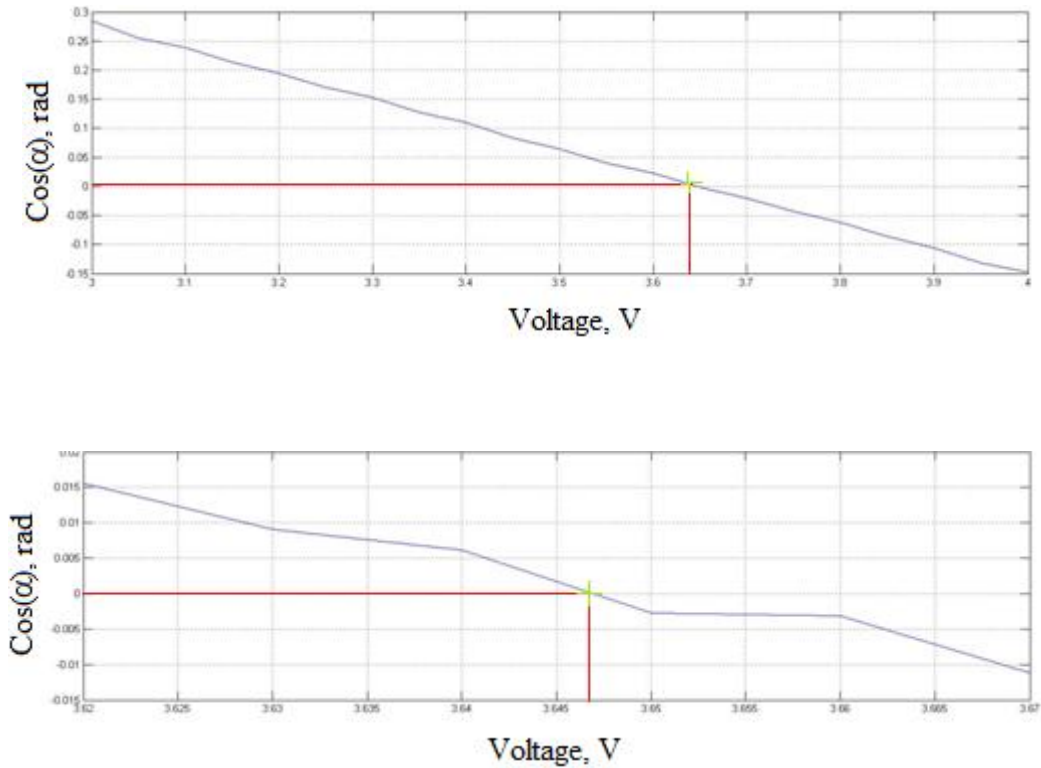


Fig. 8. Iterations of 5-step algorithm. The resulting voltages are indicated.

The last iteration has steps in between the voltages already below the resolution of the 12-bit DAC system. This shows the convergence of the algorithm to voltage that will produce a displacement corresponding to exactly $\frac{\pi}{4}$ phase shift in the path of the reference beam.

Once the system has been calibrated it was tested for accuracy and resolution. A calibration flat plate with flatness in the pico-meter range was used for testing of the Linnik interferometry setup. Fig. 9 is a histogram showing the readings across the surface of the plate:

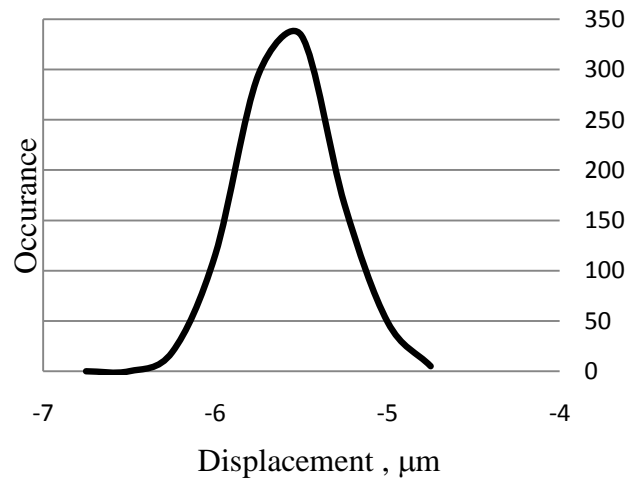


Fig. 9. Distributions of surface measurements of a flat plate.

The occurring bell curve corresponds to a standard deviation in the data of 0.25nm, thus this is the measurement resolution of the interferometric system. This shows that the calibrated system proved to be precise and stable, thus allowing us for accurate measurement of the performance of the bi-material beams.

3.2. MEMs Micro Mirror Array

Initial experimentation consisted of a proof of concept to determine the feasibility of using an interferometer for grayscale pattern generation. To test this, a sample array of pixels was required that could be predictably moved in and out of plane. An off-the-shelf micro mirror array manufactured by Boston Micromachines Corporation was purchased for testing. The chip consists of an array of 18×18 individually addressed micro mirrors; each $600\mu\text{m}$ square mirror has maximum out-of-plane displacement of approximately 400nm . The actuation of the mirrors is completely analog and can therefore be used to demonstrate the interferometers ability to measure continuous displacements. As a result, the interferometer will generate continuous grayscale patterns. Along with the micro mirror chip, a power supply to actuate the mirrors and a program to control them were provided and used for testing purposes. A description of hardware, software and testing methodology is described in Appendix D: Micro Mirrors Experimental Setup.

The calibrated Linnik interferometer was used to measure the micro mirrors in a series of experiments. By using double exposure mode in the interferometer's capture software, changes in displacement were measured. This consisted of capturing a reference position of the mirrors to which all further positions will be compared. By taking a reference with the beams at roughly half their maximum displacement, it was possible to measure both positive and negative relative out of plane displacements. As the mirror is moved up from this reference location, the optical path length of the interferometers objective beam is shortened, causing a constructive interference pattern with the reference beam. This causes the grayscale level to move toward white. As the mirror is moved down from this reference location, the optical path length of the interferometers objective beam is lengthened, causing a destructive interference pattern with the

reference beam. This causes the grayscale level to move toward black. The ability to take a new reference with the mirrors at any position is an important advantage that measuring with an interferometer has over other detection methods: its ability to measure relative change, with no regard for absolute position.

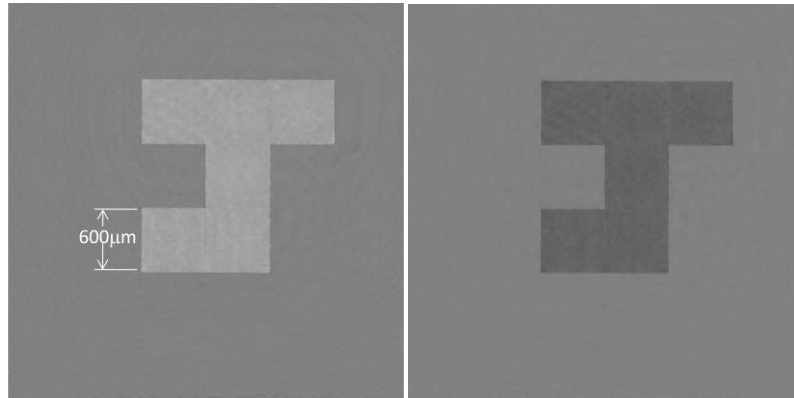


Fig. 10. MEMS generation of the letter "J", viewed using interferometry.

Fig. 10 shows images taken using the Linnik interferometer; the interferometer directly relates changes in height to varying grayscale levels. The first image shows a positive out-of-plane displacement while the second shows a negative displacement.

This experiment successfully demonstrated the interferometer's ability to generate grayscale patterns. The setup also proved capable of continuous detection; in fact the interferometer was so sensitive that it was able to detect nanometer scale distortions in the mirrors as they were actuated. Some of the mirrors in the array showed signs of fatigue or perhaps manufacturing defects. Fig. 11 shows distortions of the surface of an elevated mirror as well as distortions on the other mirrors surrounding it.

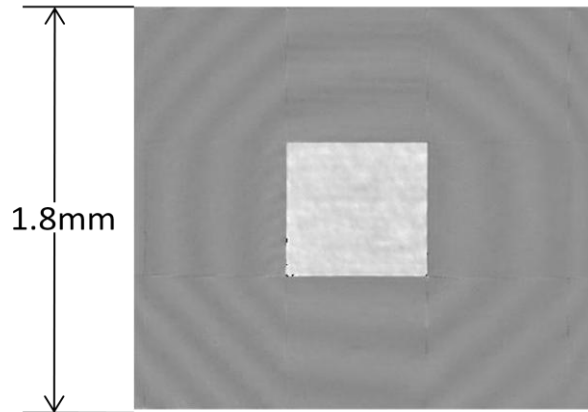


Fig. 11. Small distortions in the surface of the micro mirrors.

As the mirrors were actuated with increasing voltage, these variations seemed to increase in magnitude and frequency. Additionally, these distortions seemed to get worse over time. That is, continued use of the mirrors caused these variations to have increased magnitude. These longitudinal effects suggest that perhaps the mirrors were being stressed beyond their limits, and were subject to fatigue as a result. The octagonal distortions surrounding the elevated mirror could potentially be a result of cross talk between the actuation mechanisms. If the actuation of the mirrors is, for instance, electromagnetic, then surrounding mirrors might ‘feel’ the magnetic field of an adjacent mirror. Of course, these observations were merely qualitative and future studies are necessary to verify the accuracy of these assertions.

Successful pattern generation qualified MEMS and interferometry for use in detection of nanometer scale deflections of thermally actuated MEMS. In place of the electrically actuated micro mirror pixels, an array of thermally actuated pixels would allow for thermal imaging. MEMs bimaterial beams with a reflective coat would fit this description.

4. Realization of pattern generation using MEMS

4.1. Simple Bimaterial Cantilever Beam

4.1.1. Analytical Model

4.1.1.1. Theory, Parameters, and Calculations

An analytical model of a bimaterial cantilever beam was developed by Stephen Timoshenko in 1925. This equation determines the radius of curvature of a bimaterial, two-dimensional beam. Several parameters are required for Timoshenko's general equation (Timoshenko, 1925) found below in Equation 6; a description of these parameters and their values are listed in Table 1. A full derivation of this equation is shown in Appendix A: Derivation of Timoshenko's Analytical Model.

$$\rho := \frac{h \cdot \left[3(1 + t_m)^2 + (1 + t_m \cdot E_n) \cdot \left(t_m^2 + \frac{1}{t_m \cdot E_n} \right) \right]}{6 \cdot (1 + t_m)^2 \cdot (\alpha_{\text{Gold}} - \alpha_{\text{Si}}) \cdot \Delta T} \quad \text{Equation 6}$$

$$\rho = h(3(1 + m)^2)$$

Table 1. Beam parameters.

Parameter	Symbol	Value
Thickness - Material One	t_1	120nm
Thickness - Material Two	t_2	600nm
Ratio of t_1/t_2	t_m	0.2
Young's Modulus -Material One	E_1	77GPa
Young's Modulus - Material Two	E_2	180GPa
Ratio of E_1/E_2	E_n	0.428
CTE -Material One	α_1	14.2ppm/K
CTE - Material Two	α_2	0.8ppm/K
Beam Length	l	64 μ m

This yields a radius of curvature of $\rho = 36.459\text{mm}$. Using the length of the beam as an arc about a circle with radius ρ , the angle θ can be calculated; Fig. 12. The vertical displacement δ can then be calculated as $\delta := \rho - \rho \cdot \cos(\theta)$.

There are limitations to Timoshenko's general model; the equations make no mention of the width of the beam, and so disregard any motions along that axis. Realistically, the same stresses that cause

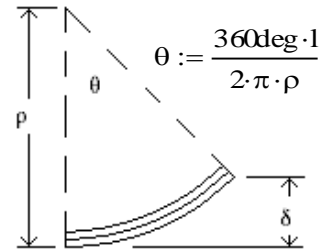


Fig. 12. Graphical representation of Timoshenko's solution.

curvature along the length of the beam will cause curvature along the width. This effect can be taken into account by entering Poisson's

Ratio into the equations. More detailed calculations can be found in Appendix B: Analytical Calculations of a Bimaterial. For the parameters listed above, the deflection of a bimaterial

beam can be calculated as $23.797 \frac{\text{nm}}{\text{K}}$.

4.1.1.2. Uncertainty Analysis

Conducting uncertainty analysis on the analytical model allowed for the determination of an appropriate thermal resolution for any experimental testing that would follow. Using the R.S.S. method of uncertainty analysis the percent error contribution of all relevant parameters was determined, refer to Appendix B: Analytical Calculations of a Bimaterial for a full description of the calculations. The MEMs micro cantilever beam array was manufactured using the surface micro machining technique, a precise manufacturing process; as such, it was determined that an appropriate approximation for the uncertainty in the physical parameters of the beams was 1%.

Temperature, being the only parameter in the deflection of the beams that we had control over, became the focus of these calculations. Initially we used an uncertainty in temperature of $\delta T = 0.5\text{K}$. With this uncertainty, temperature contributes 99.7% of the error in the analytical

model. By repeating the process using different uncertainties in temperature, an appropriate value was found. This calculation determined that in order for the error in temperature to have a minimal effect on the error of the system, an uncertainty of 0.01K was required. This value was used later when designing an experimental setup to control the temperature of the experimental array. The calculated values for percent error contribution are shown in Table 2.

Table 2. Percent error contribution.

Parameter	Percent error contribution		
	$\delta T = 0.5K$	$\delta T = 0.05K$	$\delta T = 0.01K$
t_1 Thickness of layer 1	0.02	1.89	8.61
t_2 Thickness of layer 2	0.12	10.01	46.00
E_1 Elastic modulus of layer 1	0.02	1.53	6.95
E_2 Elastic modulus of layer 2	0.02	1.53	6.95
α_1 Coefficient of thermal expansion of layer 1	0.05	3.65	16.63
α_2 Coefficient of thermal expansion of layer 2	0.00	0.01	0.05
T Temperature	99.77	81.29	14.81

4.1.2. Computational Model

4.1.2.1. Choosing an Element Type

Shell theory relies on a simplification of geometry to a zero-thickness surface. It does, however, use thickness dimensions for determination of resistance to bending and stretching - two characteristics that are vital to shell theory. This approach to FEM is favorable when attempting to model thin films or geometry with a very small aspect ratio, which can be defined as the ratio of width and length dimensions to the thickness dimension. For the models discussed below, deformations in the thickness direction are not of note. Rather, the deformations and deflections along the length and width of the beam are important, and discretization in those dimensions is necessary.

For the FEM model described below, ANSYS11 was used, and the *SHELL99: Linear Layered Structural Shell* element type was chosen. This element type allows up to 250 structural shell layers. Each element has eight nodes with six degrees of freedom: translation and rotation about the x-, y-, and z-axis. The model assumes no slippage between layers, linear thermal gradients, and that the stress varies linearly through the thickness dimension. SHELL99 is not convergent in large displacement scenarios. The model described in this section fits all these assumptions.

4.1.2.2. Determining the Element Size

The mesh size of the FEM model was validated by iteratively reducing the element edge length of the FEM model until the desired accuracy could be obtained. The results of this test are listed below in Table 3. Fig. 13 shows the FEM model in its meshed state. Note the right end of the beam where the end constraints are shown. The right-most edge of the beam is fixed on all degrees of freedom.

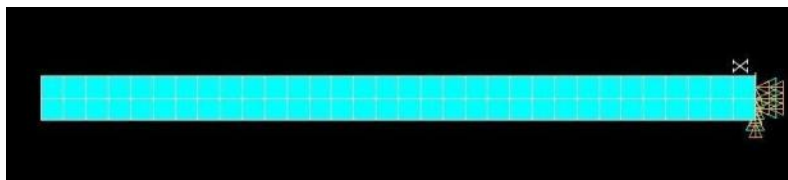


Fig. 13. Meshed cantilever beam showing end constraints.

The deflection converges to the nanometer with a mesh edge length of around $2\mu\text{m}$. Meshes of smaller edge length only affect the beam on the picometer scale. *Note: The information shown in Table 3 is for the beam geometry described in the MEMS Device Bimaterial Beam Array section on page - 26 -. Similar methodology was used to determine the necessary mesh size for the simple cantilever described in this section.*

Table 3. Determination of mesh size.

Mesh Edge Length Determination		
Edge Length (μm)	Deflection (μm)	Difference (μm)
10	-0.106341	-7.408E-03
5	-0.105921	4.200E-04
2	-0.105866	5.500E-05
1	-0.105853	1.300E-05
0.5	-0.105845	8.000E-06
0.25	-0.105844	1.000E-06

4.1.2.3. Results and Validation

The computational model of the simple cantilever beam yielded a result of 23.978nm/K.

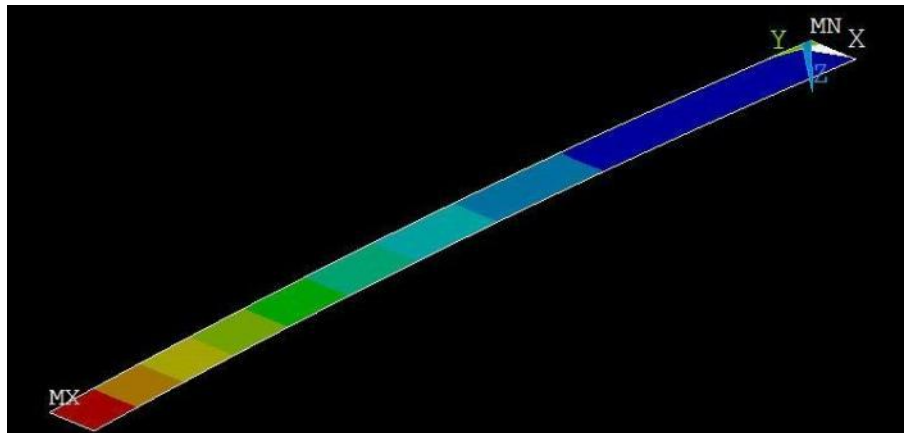


Fig. 14. Isometric view of FEM solution for simple cantilever.

Additionally, the thermal response of the beams according to both analytical and computational models is completely linear. Naturally, with greater temperature differences the lines diverge slightly. A graph of the deflection vs. relative temperature can be seen in Fig. 15.

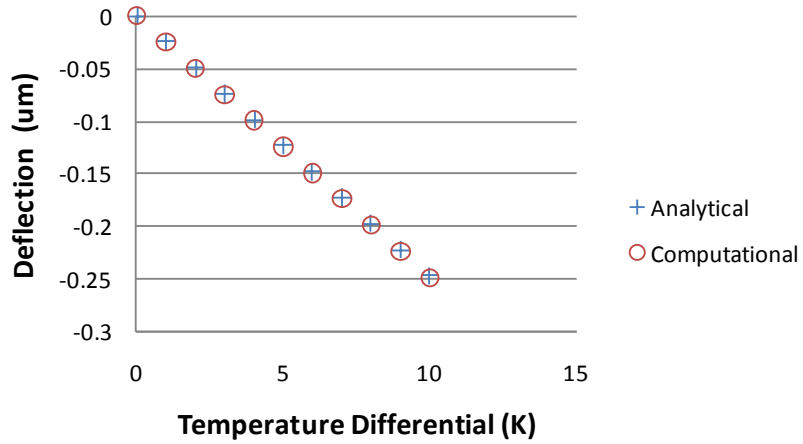


Fig. 15. Analytical and computational model comparison.

In order to validate its accuracy, the FEM model had to be compared against the analytical solution described previously. Recalling from that section, Timoshenko's model yielded a sensitivity of 23.797nm/K. With less than a percent error between the two solutions, the FEM model can be confidently altered to model more complex geometry.

4.2. MEMS Device Bimaterial Beam Array

4.2.1. MEMS Device Fabrication Process

The MEMS devices described in this section used the fabrication process discussed in the Surface Micromachining section of this paper. A step by step illustration is shown in Fig. 16. The process begins with a partially masked silicon wafer. These masks prohibit chemical etchant from attacking the silicon in these areas (refer to Fig. 16). Next, the sample goes through a wet etching process that removes material from the silicon wafer. The second step is a chemical vapor deposition process where a SiO₂ sacrificial layer is deposited. This sacrificial layer will later be removed. The sample then goes through a chemical polishing process whereby the masking material is removed and some of the sacrificial layer is removed until the original silicon wafer is flush with the SiO₂. The fourth step deposits the structural bimaterial layers so that one end of each layer is attached to the posts left on the silicon wafer. The rest of the bimaterial lies on top of the sacrificial layer, which is then removed in a wet etching process. The removal of this layer frees the individual beams so that they may deflect as a result of temperature change.

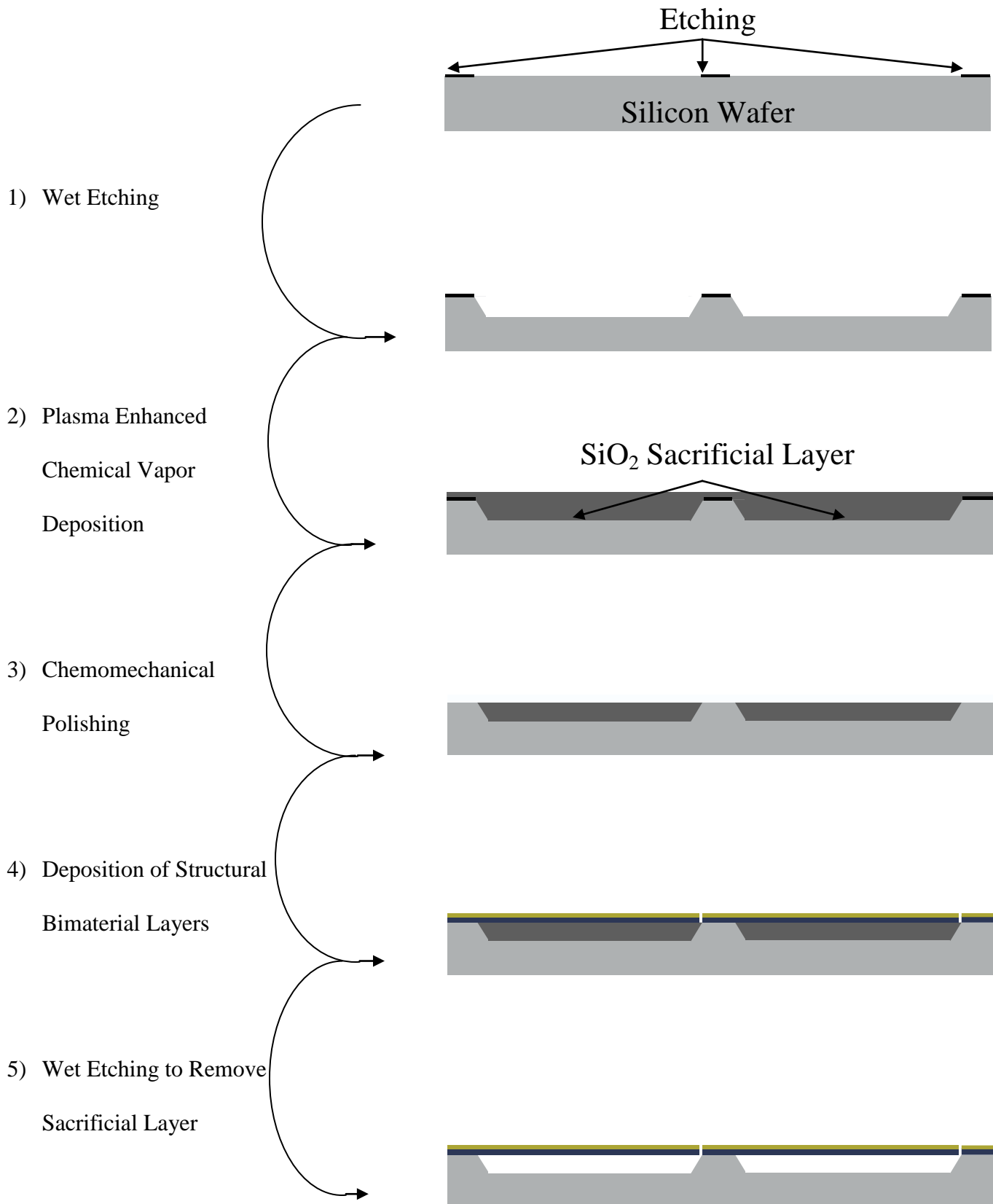


Fig. 16. Fabrication process used for bimaterial micro cantilever beams array.

4.2.2. Computational Model

The aforementioned computational model of a simple cantilever beam validated that the FEM model could be reliably expanded to predict the behavior of beams with more complex shapes, such as the ones to which the Experimental Results section on page - 31 - refers. A highly magnified image of these beams is shown in Fig. 17. Note the presence of release holes, a trademark of the surface micromachining process.

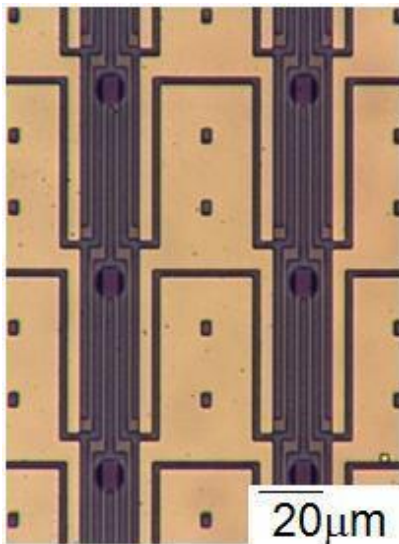


Fig. 17. 50x magnified picture of bimaterial cantilever beam array.

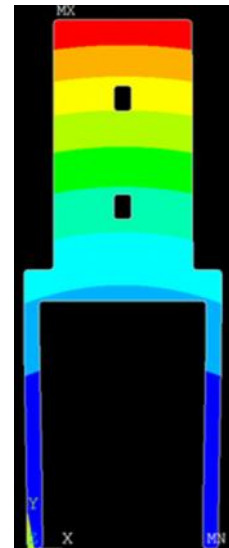


Fig. 18. Graphic representation of bimaterial cantilever beam from FEM model.

During the removal of sacrificial layers, etchant is allowed to pass through these holes so that the sacrificial material may be easily removed. The image shown in Fig. 18 has the release holes (and corner fillets) modeled, although the FEM model does not take these into consideration. These features affected the computation by only 1-3 picometers; these effects were deemed negligible.

4.2.2.1. Beam Response

Graphic results are shown for the FEM model of the bimaterial strip below. Fig. 19 shows the pixel in its meshed state. Note the end constraints on the legs of the beam. These edges are constrained from all degrees of freedom. Fig. 20 shows an isometric view of the results of the FEM model.

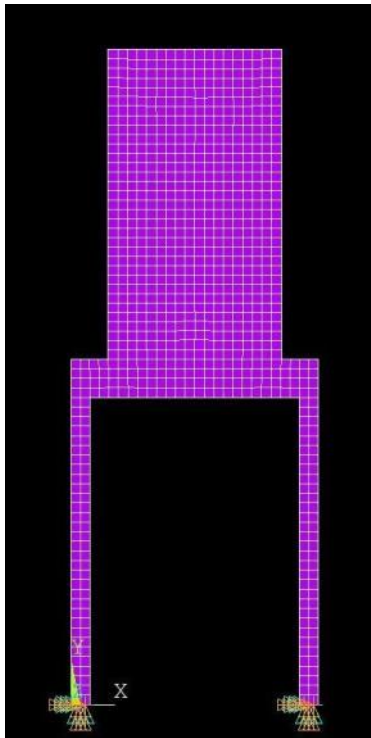


Fig. 19. Meshed bimaterial pixel showing end constraints.

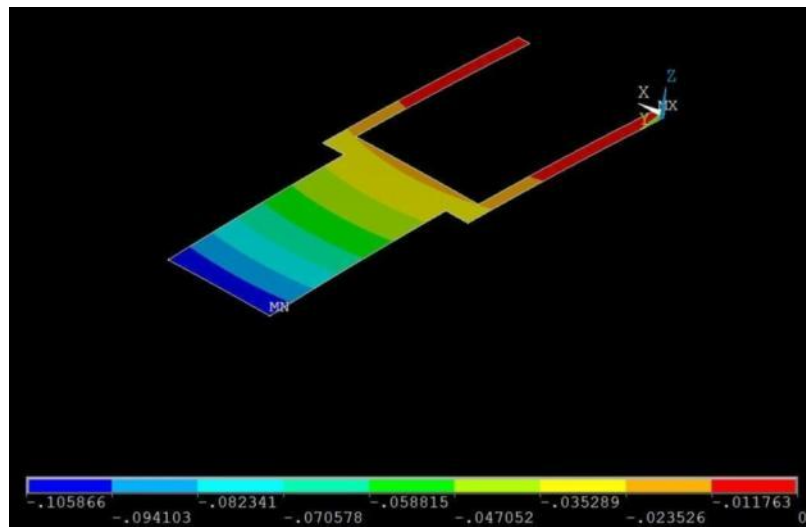


Fig. 20. Isometric view of bimaterial pixel.

Table 4 shows the deflection of the beam shown in Fig. 18 as a function of temperature difference. Clearly, there response of the beams is linear. This relationship is shown graphically in Fig. 21.

Table 4. Beam deflection vs. temperature differential.

Temperature Difference (k)	Deflection (μm)	Difference
1	-0.105853	-
2	-0.211706	-0.105853
3	-0.317558	-0.105852
4	-0.423411	-0.105853
5	-0.529264	-0.105853
6	-0.635117	-0.105853
7	-0.740970	-0.105853
8	-0.846822	-0.105852
9	-0.952675	-0.105853
10	-1.058528	-0.105853

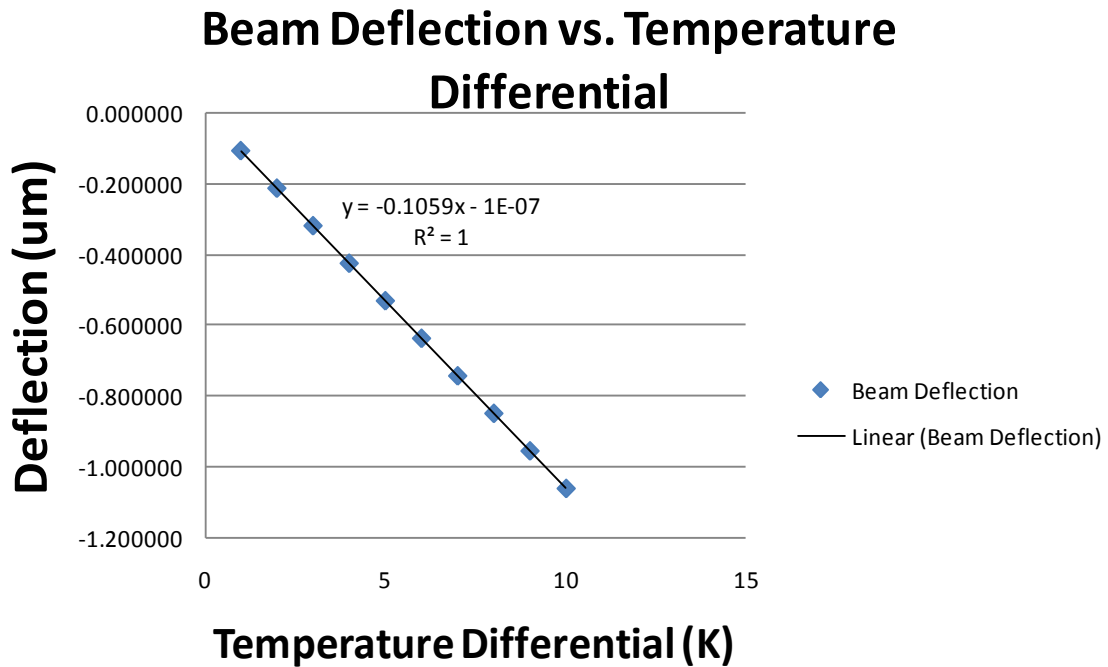


Fig. 21. Beam deflection vs. temperature with linear best fit.

4.3. Experimental Results

Experimental data was collected using the Linnik interferometer; the data was processed and analyzed using a collection of software developed by Professor Cosme Furlong. Raw data was first masked using a MATLAB program to avoid errors when using the unwrapping software. The raw masked data was then phase-unwrapped using a fluid unwrapping algorithm. Finally, unwrapped data was exported to MATLAB for analysis; with this data, both absolute and relative position could be determined. By relating these displacements to the temperature of the beams at the time of the experiment and taking several readings at different temperatures, the sensitivity of the beams was calculated. Then, by combining the thermal sensitivity of the array with the resolution of the interferometer, the thermal resolution of the system was determined.

A thermal loading station was designed and built to accurately determine the temperature of the experimental array. Fig. 22 shows a CAD model of this setup while Fig. 23 shows a realization of this design. Two TECs were used to generate a temperature difference between the base plate, a piece of eighth inch aluminum, and a heat sync fastened below it. There were two major considerations in the design of this setup; first, that the experimental array is to be kept as stationary as possible in the heating and cooling process, and second, that the two thermal bodies be thermally isolated to ensure thermal stability.

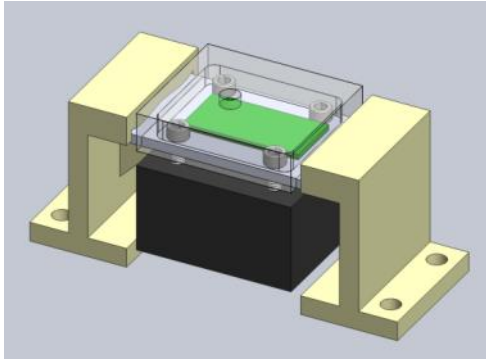


Fig. 22. CAD model of thermal loading station.



Fig. 23. Design realization of thermal loading station.

Initial setups were fastened from the bottom up, and the expansion of components as heat was applied caused motion in the array. By grounding the setup from the top plate, any expansion in the components as a result of thermal loading should have little effect on the position of the plate. In addition, the clamps and screws used in the setup are all plastic, to ensure that the plate and the heat sink are thermally isolated. A plastic cover with a small hole for the laser is placed over the experimental array. This cover is designed to reduce the effects of convection to the surrounding air.

Using the RSS method of uncertainty analysis, it was determined the uncertainty in the temperature should be 0.01K. Rather than try to control the temperature of the setup to a high precision (a daunting task), it was easier to insert two high sensitivity thermistors into the top plate to measure the temperature to a high degree of accuracy. These calculations are shown in the Uncertainty Analysis section of this paper on page - 21 -. For all experiments thermal grease was used between all surfaces to maximize the thermal response of the system.

Data sets were collected by first capturing a reference image at an initial temperature and then taking ten images at even intervals as the array was either heated or cooled. Data sets were taken at a variety of starting temperatures, both heating and cooling, and with the ten images

captured in temperature ranges from 1°C to 5°C. By using these test conditions, any non-linearity or hysteresis in the system should have been discovered.

After obtaining the raw experimental data it was noticed that directly applying unwrapping algorithm to it resulted in errors mainly because of data noise at the edges of the beams. The noise was due to irregular illumination of the edge areas of the beams and since the displacement measurements are based on the illumination readings, this phenomenon caused errors in reading the surface of the beams. This forced us to create a MATLAB program than reads the raw data and automatically masks it so that only the surface of the beams is visible along with some of the surface of the substrate. The reason why the some of the substrate is left unmasked is because of the unwrapping algorithm needs a reference position to start from so that it can calculate the displacement of the surface of the beams based on the position of the substrate, thus assuming that the substrate's surface is dimensionally stable at the test temperatures. The flowchart and coding are shown in Appendix C: . The following is an overview of the masking algorithm.

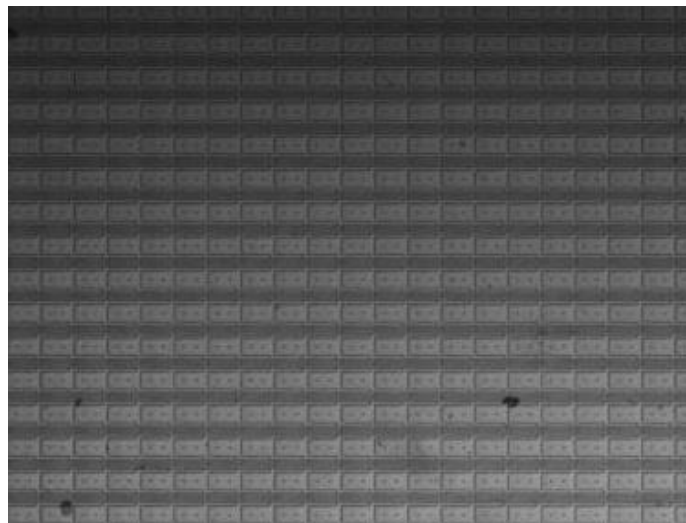


Fig. 24. Step 1 of masking algorithm.

In the first step, the program reads a file containing the image of the part of the array of beams. An example of that is shown in Fig. 24.

In the second step, the program applies an edge detection algorithm based on a Canny filter. This results in finding the borders of the beams as shown in Fig. 25. Due to the complexity of the structure of the beams we could calibrate the algorithm to detect only some of the beams. However, since the beams are all identical in shape and their position is known, we can extrapolate the information for one single beam to create a mask for all beams.

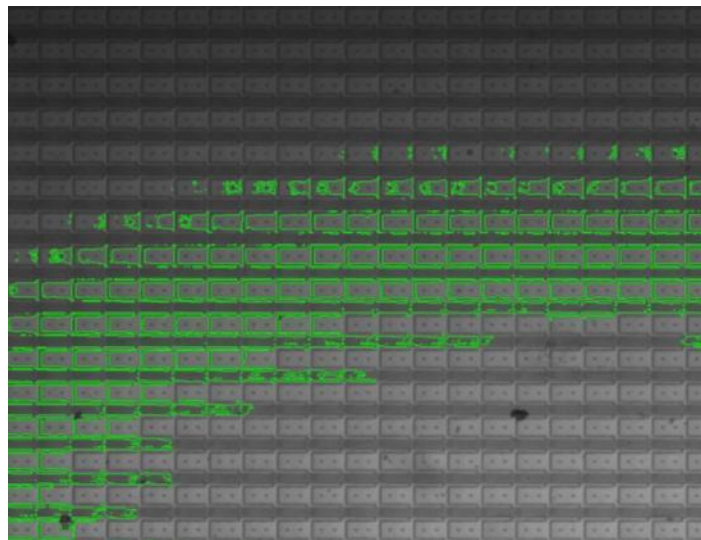


Fig. 25. Step 2 of masking algorithm.

In the third step, the program will automatically extract the shapes of the most readable beams and mask everything else, as shown in Fig. 26. At that point, the program asks the user to choose the beam that has been masked most properly. The mask for that particular beam will be later

used to mask all the other beams. This step could be further automated so that the program can make the required choice.



Fig. 26. Step 3 of masking algorithm.

In the fourth step, the mask of the beam chosen by the user is extended to include the substrate and as well a small bridge that will connect the beam with its neighbor, as shown in Fig. 27. The reasoning behind this extension is that the unwrapping algorithm needs to have access to the substrate and the surface of the beam at the same time so that it can use the substrate as a reference for the position of the beam.



Fig. 27. Step 4 of masking algorithm.

In the final fifth step, the program applies the mask to all beams thus creating a large complex mask that is applied to the whole visible part of the beam array, as shown in Fig. 28.

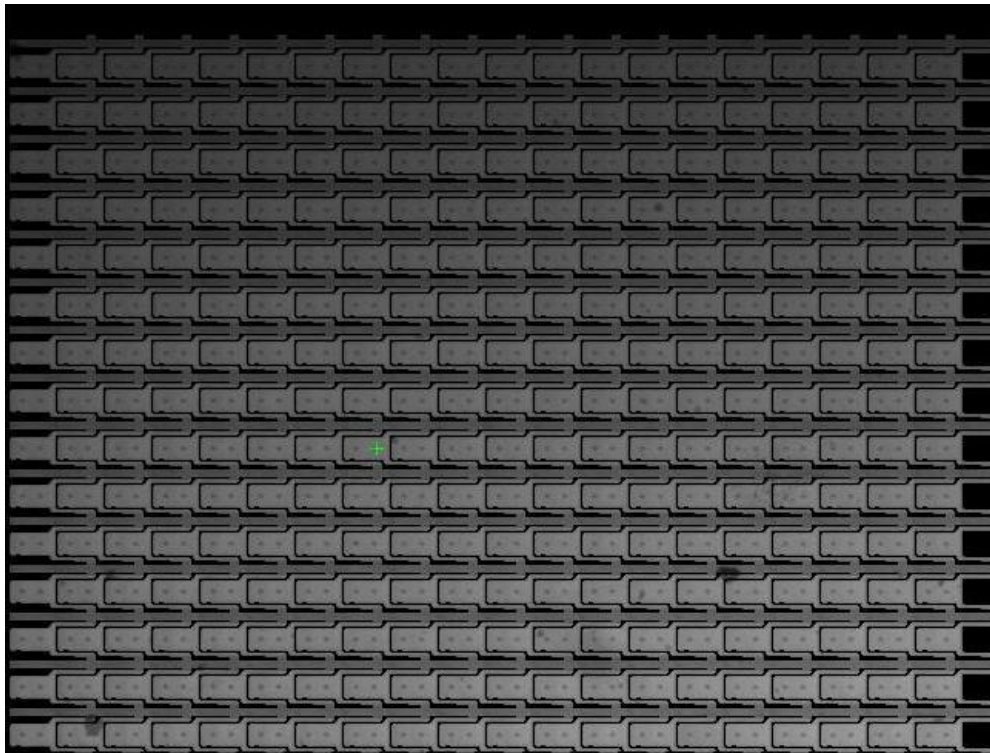


Fig. 28. Step 5 of masking algorithm.

The mask greatly reduced the noise in the data and thus allowing for the unwrapping algorithm to produce much smoother surface representing the surface of the beam. Fig. 29 shows the masked raw data and the final result of the unwrapping algorithm is shown in Fig. 30.

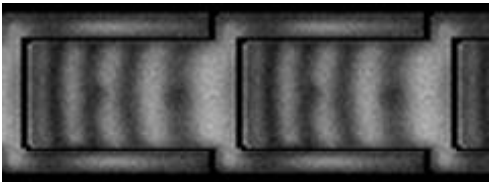


Fig. 29. Raw masked data.



Fig. 30. Phase unwrapped data collected using the interferometer.

The unwrapping algorithm produces a continuous grayscale image so that the grayscale intensity of each pixel corresponds directly to the displacement of the beam at that point. This information was read with another MATLAB program that interprets the grayscale data as displacement and generates a corresponding 3D image of the beams actual surface as read by the interferometer.

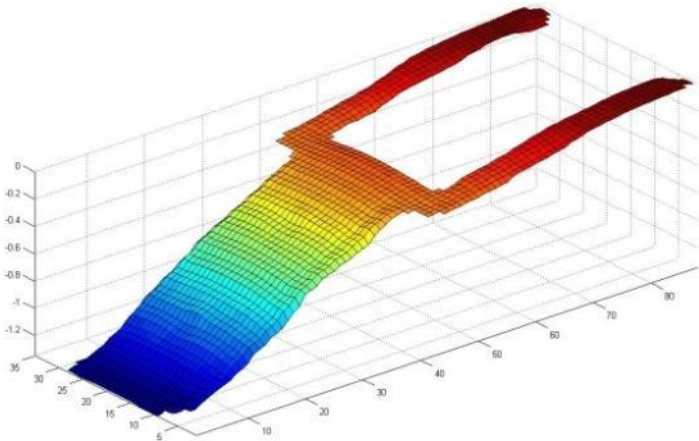


Fig. 31. A 3D view of the surface of a single beam.

Fig. 32 shows a set of raw data pictures along with the corresponding 3D surface as read by our MATLAB program. The pictures exemplify the response of the system to changes in temperature. In this case the rightmost pictures correspond to the highest temperature, thus qualitatively showing that the beams are performing as predicted by the general bi-material theory – bending down as temperature increases:

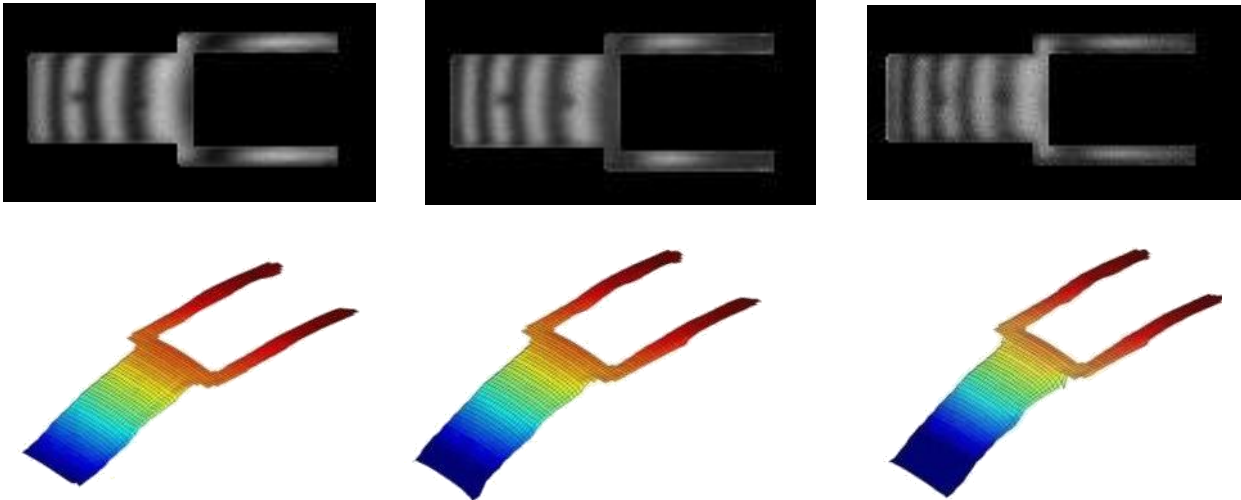


Fig. 32. Raw data at increasing temperature and corresponding experimental deflections.

Based on this information, the curvature of the beam was extracted by tracing a line on the 3D surface of the beams. This technique was applied for the extraction of the curvature of the beam at several temperatures, thus generating an array of surfaces as the one shown in Fig. 33. The flowchart and coding algorithm is shown in Appendix C: .

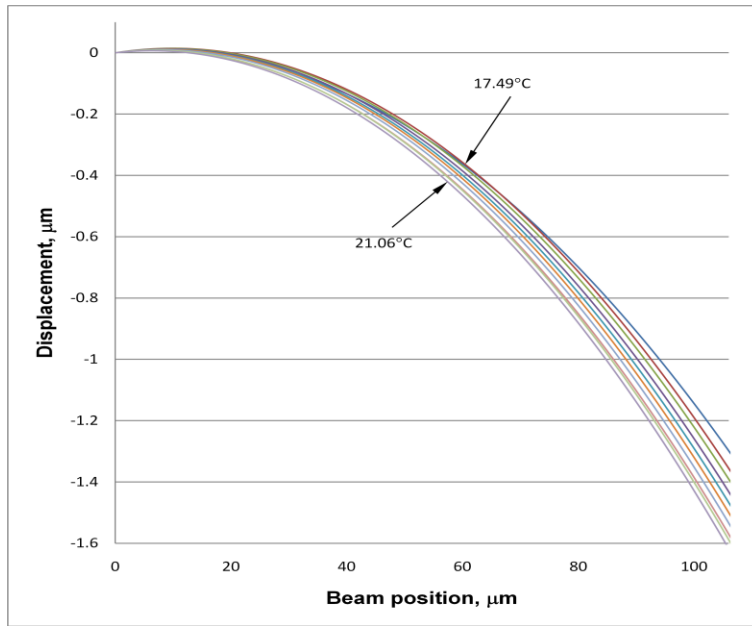


Fig. 33. Beam surfaces at varying temperatures.

By reading the position of the surface at the tip of the beam, a data for the sensitivity of the beams can be extracted. A graph of the beams' tip displacement with respect to temperature is shown in Fig. 34.

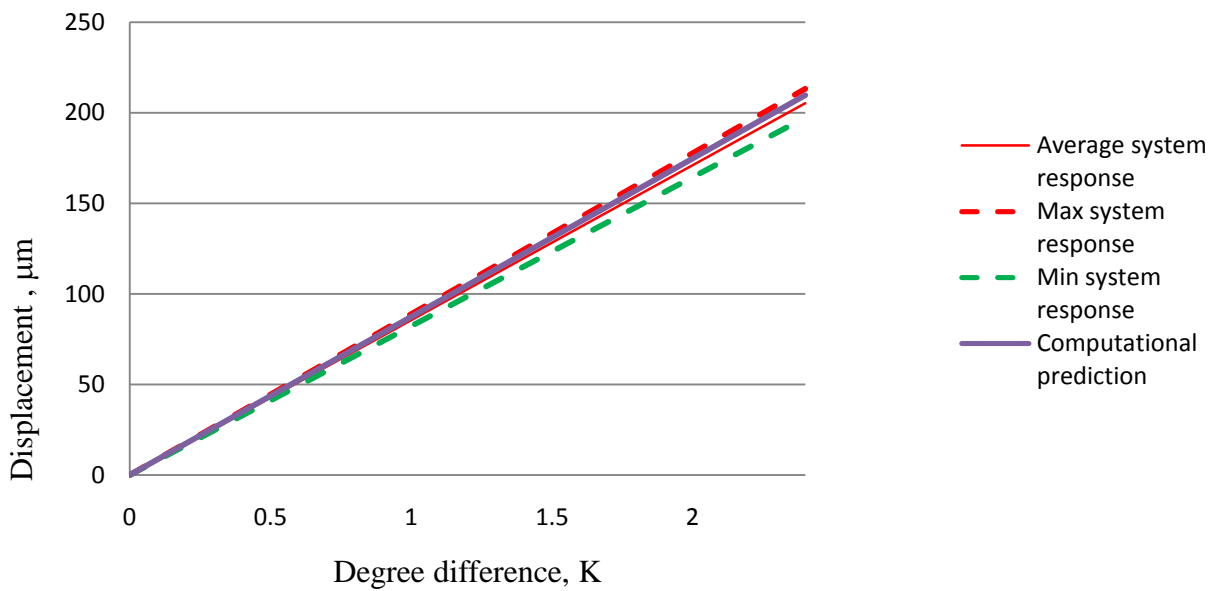


Fig. 34. Beam sensitivity comparison for FEM and experimental results.

Fig. 35 displays a zoomed-in view of the end of the above graph that shows a magnified image of the sensitivity curve of the beams:

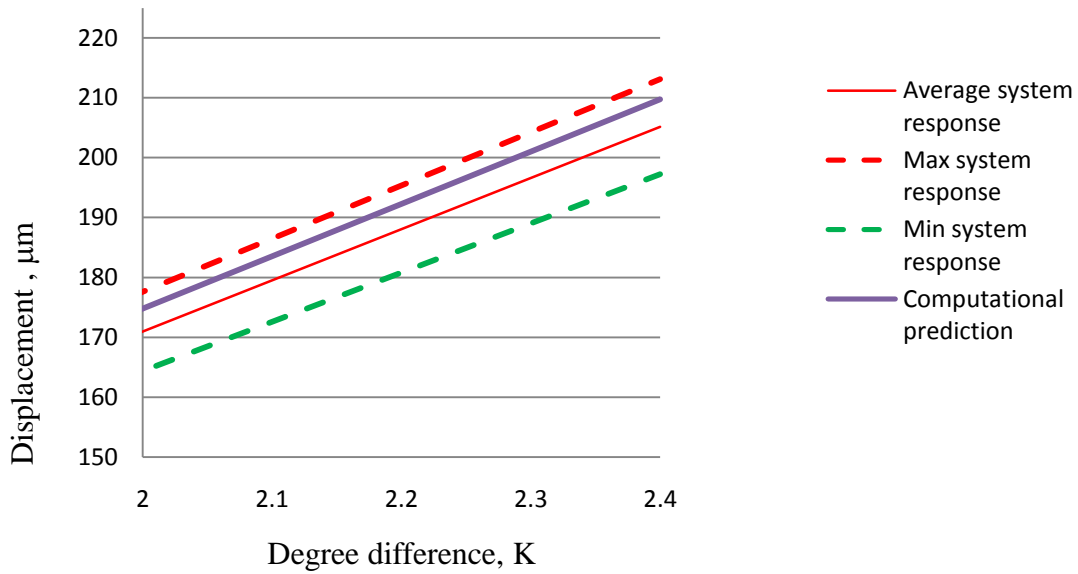


Fig. 35: Zoomed-in view of Fig. 34

It can be clearly seen that computationally predicted thermal response of the beams is within the error margin of the experimentally predicted response. A comparison of the FEM and experimental results are shown in Table 5.

Table 5. Comparison of experimental results with computational solution.

Model Type	Tip Deflection
FEM	87.39nm/K
Experimental	85.49nm/K

The error between the computational and the experimental model is less than 2.2%. This gave us confidence that our experimental procedures were done with high degree of accuracy. Having the sensitivity of the MEMS bi-material beams array and the sensitivity of the interferometric measurement system, the resulting overall thermal resolution of the system is 3mK. By comparing this value to the resolution of existing high-end infrared imaging systems, it can be seen from Table 6 that our system achieves more than 800% increase in resolution.

Table 6. Comparison of thermal resolutions.

Infrared imaging system	Resolution
Critical Imaging IAC-580 SurveyIR	30mK
FLIR Systems ThermoVision® SC8000	25mK
Our Interferometric setup	~3mK

5. Conclusions and Recommendations

5.1. Parametric Investigation and Recommendations

Fig. 36 shows, according to Timoshenko's equations, the deflection of a bimaterial beam with the properties listed in Table 1, except for the thicknesses of the bimaternal which are varied along the X-axis. Clearly, the shape of the curve shows a local maximum and possible point for thickness ratio optimization.

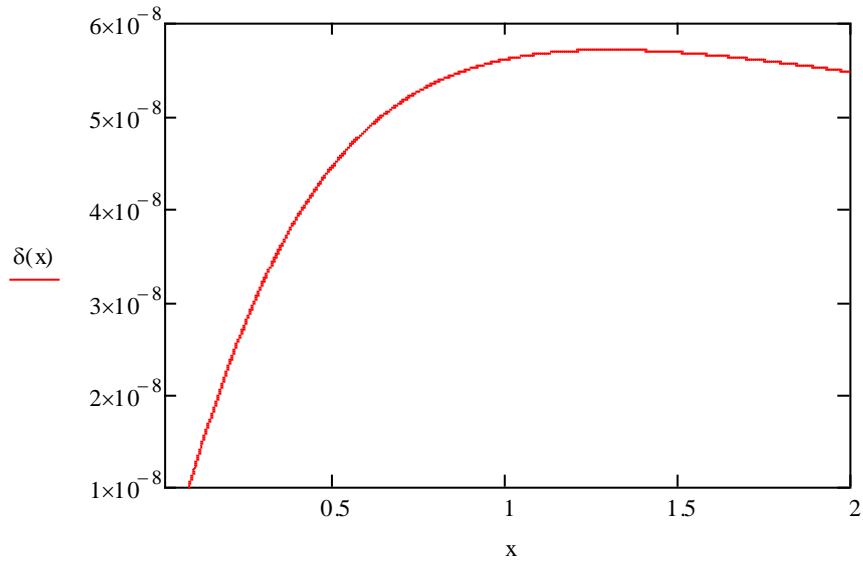


Fig. 36. Deflection vs. thickness ratio from analytical solution.

Referring to Fig. 37, FEM was used to determine the effects of thickness ratio. Showing the same curve as Fig. 36, the FEM also shows a local maximum. Note the error bars. If these bars represent tolerances in the layer deposition process upon which surface micromachining relies, the thickness ratio of a single bimaterial beam can differ from the expected result. However, this poses more of a design problem at some areas along the curve. More specifically, for a thickness ratio along the steep side of the curve, a small variance in each layer's thickness may provide a very different deflection than what is expected by the designers. In contrast, if the designers of the bimaterial beam choose a thickness ratio corresponding to the flat region of the curve, a much smaller difference in deflection will occur.

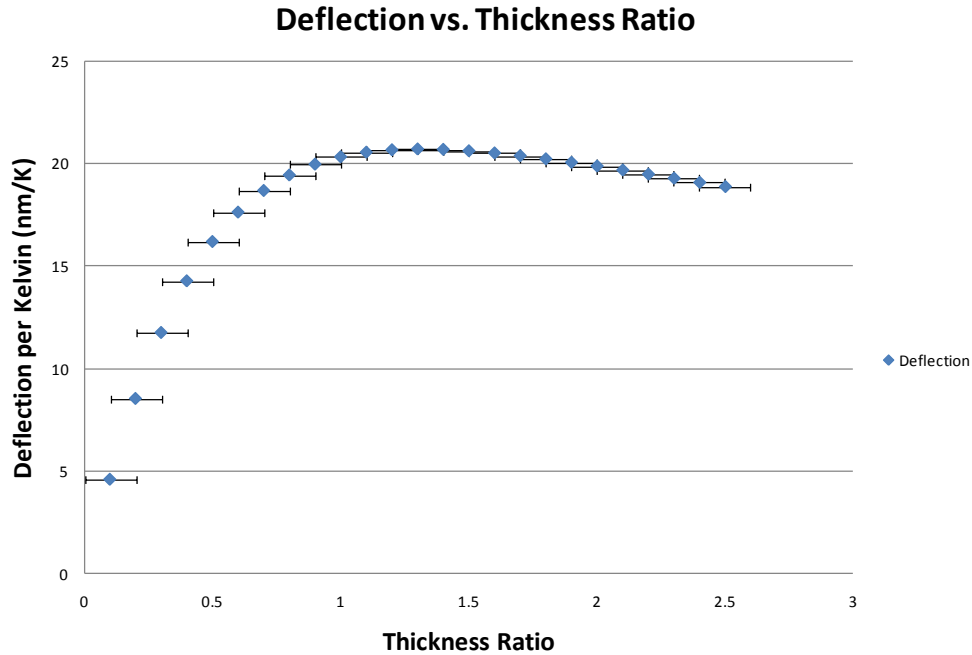


Fig. 37. Deflection vs. thickness ratio from FEM model.

Interestingly, overall beam thickness should also be carefully chosen. Aside from the obvious structural issues presented by too thin or too thick a beam, the techniques of layer deposition have an additive effect on the manufacturing error. Assuming that each deposited layer is thicker than intended, the more layers, the larger the error will be. Since determining the exact geometry of the beams (after they're manufactured) is very difficult and usually destructive, great care should be taken when designing bimaterial MEMs devices.

Additionally, the effect of varying the ratio of elastic modulus of the two materials was computed. In this case, it is clear that as the ratio increases, so does the sensitivity of the beam. That is, if the elastic modulus of the top layer is higher than that of the lower layer, the sensitivity will be higher.

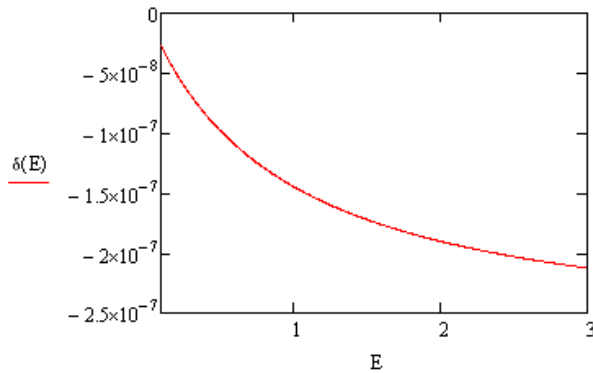


Fig. 38. Elastic modulus ratio vs. deflection.

5.2. Conclusions and future work

In our project we successfully demonstrated that interferometry can be used for real time gray-scale imaging. We also managed to characterize MEMS thermally actuated bimaterial beam array. With our analytical and FEM model of the thermal sensor we allow for future design optimization of the beams array for specific applications. Overall, our system demonstrated a thermal resolution not achievable by any currently available conventional commercial technologies. Future work should be focused on miniaturization and full system integration of the system. Further investigations on achieving full system mobility should be conducted.

6. References

ANSYS Inc., *Ansys User's guide v 9.0*, Canonsburg, PA, 2007.

Boston Micromachines, Inc. <<http://www.bostonmicromachines.com/>>

Critical Imaging, Inc., retrieved April 2009 <www.criticalimaging.net>

Flir Thermal Imaging, Inc, retrieved April 2009 <www.flir.com>

GW INSTEK. Accessed 31 August 2008. < <http://www.goodwill.com.tw/html/en/index-e.asp> >

Hibler, R.C.. 2005, *Mechanics of Materials*, 6th ed. Pearson, Upper Saddle River, NJ.

Moaveni, S., 2003, *Finite Element Analysis: Theory and Application with ANSYS*, Pearson, Upper Saddle River, NJ.

National Instruments. "Laser Interferometer – Developer Zone" Accessed 15 September 2008. <<http://zone.ni.com/devzone/cda/ph/p/id/126/>>.

Norton, R.L., 2006, *Machine Design: An Integrated Approach*, Pearson, Upper Saddle River, NJ.

Pal, Sagnik; kemiao, Jia; Xie, Huikai. *An Electrothermal Mirror with High Linear Scanning Efficiency*. University of Florida, Gainesville.

Pelesko, J. A, and Bernstein, D. A., *Modeling MEMS and NEMS*, CRC Press, Boca Raton, FL, 2003.

S. Timoshenko, S., "Analysis of bi-metal thermostats," *J. Opt. Soc. Am.*, 11:233-255, 1925.

Thorlabs, Inc., "Piezo-Electric Actuators," Accessed 24 February 2009. <<http://www.Thorlabs.com>>

T-R. Hsu, *MEMS & Microsystems: design and manufacture*, McGraw-Hill, NY, 2002.

Appendix A: Derivation of Timoshenko's Analytical Model

A derivation of the analytical model used is as follows:

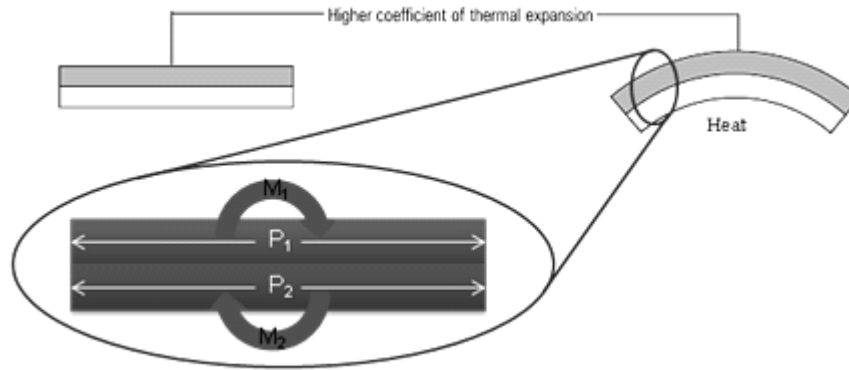


Fig. 39. Bimaterial beam stress element.

From static analysis we can determine that:

$$P_1 = P_2 = P, \quad (A1)$$

$$\frac{P * h}{2} = M_1 + M_2, \quad (A2)$$

$$M_1 = \frac{E_1 I_1}{\rho}, \quad M_2 = \frac{E_2 I_2}{\rho}, \quad (A3)$$

And therefore:

$$\frac{P * h}{2} = \frac{E_1 I_1}{\rho} + \frac{E_2 I_2}{\rho}. \quad (A4)$$

On the bearing surface the elongation of the two layers must be equal, therefore;

$$\alpha_1(t - t_0) + \frac{P_1}{E_1 \alpha_1} + \frac{\alpha_1}{2\rho} = \alpha_2(t - t_0) + \frac{P_2}{E_2 \alpha_2} + \frac{\alpha_2}{2\rho}. \quad (A5)$$

Combining equations A5 and A4 yields

$$\frac{h}{2\rho} + \frac{2(E_1 I_1 + E_2 I_2)}{h * \rho} \left(\frac{1}{E_1 \alpha_1} + \frac{1}{E_2 \alpha_2} \right) = (\alpha_2 - \alpha_1)(t - t_0). \quad (\text{A6})$$

Letting,

$$\frac{\alpha_1}{\alpha_2} = m \quad (\text{A7})$$

and

$$\frac{E_1}{E_2} = n \quad (\text{A8})$$

A correction factor is needed to account for the Poisson's effect on the beams. In place of equation A8 we used

$$n = \frac{E_1(1 - \nu_2)}{E_2(1 - \nu_1)}. \quad (\text{A9})$$

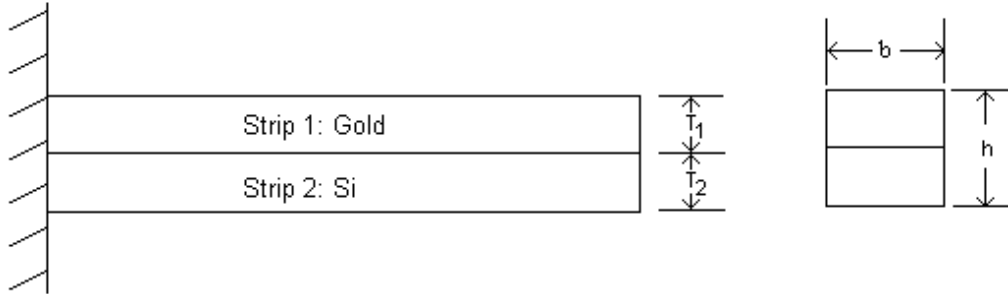
Combining equations A9, A7, and A6 yields

$$\frac{1}{\rho} = \frac{6(\alpha_2 - \alpha_1)(t - t_0)(1 + m)^2}{h \left(3(1 + m)^2 + (1 + m * n) \left(m^2 + \frac{1}{m * n} \right) \right)}. \quad (\text{A10})$$

Once the radius of curvature, ρ , of the beam is known, the displacement can easily be calculated using the length of the beam.

Appendix B: Analytical Calculations of a Bimaterial

Analytical method for determining vertical displacement in bimetallic cantilever beams



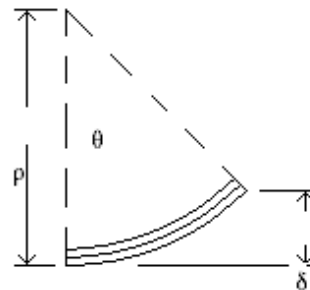
Young's modulus:	Coefficient of thermal expansion:	Beam Dimensions
$E_1 := 77\text{GPa} = 7.7 \times 10^4 \cdot \frac{\text{kg}}{\mu\text{m} \cdot \text{s}^2}$	$\alpha_1 := 14.2 \cdot 10^{-6} \cdot \frac{1}{\text{K}}$	$t_1 := 120\text{nm}$
$E_2 := 180\text{GPa} = 1.8 \times 10^5 \cdot \frac{\text{kg}}{\mu\text{m} \cdot \text{s}^2}$	$\alpha_2 := 0.8 \cdot 10^{-6} \cdot \frac{1}{\text{K}}$	$t_2 := 600\text{nm}$
$E_n := \frac{E_1}{E_2}$	$t_m := \frac{t_1}{t_2}$	$h := t_1 + t_2$
	$\Delta T := 1\text{K}$	$l := 136.5\mu\text{m}$

Radius of curvature

$$\rho := \frac{h \cdot \left[3(1 + t_m)^2 + (1 + t_m \cdot E_n) \cdot \left(t_m^2 + \frac{1}{t_m \cdot E_n} \right) \right]}{6 \cdot (1 + t_m)^2 \cdot (\alpha_2 - \alpha_1) \cdot \Delta T} = -0.106 \text{ m}$$

The angle θ can be calculated:

$$\theta := \frac{360 \cdot \text{deg} \cdot 1}{2 \cdot \pi \cdot \rho} = -0.074 \cdot \text{deg}$$



The movement of the free end of the beam can be calculated as:

$$\delta := \rho - \rho \cdot \cos(\theta)$$

$$\delta = -87.852 \cdot \text{nm}$$

Sagnik Pál and Huikai Xie

$$\beta := \frac{6 \cdot (1 + t_m)^2}{\frac{1}{E_n \cdot t_m} + E_n \cdot t_m^3 + 2(2t_m^2 + 3 \cdot t_m + 2)}$$

$$\theta_c := \beta \cdot \frac{1}{h} \cdot (\alpha_2 - \alpha_1) \cdot \Delta T = -0.074 \cdot \text{deg}$$

Error Analysis

$$\begin{aligned} \delta t_1 &:= 0.01 \cdot t_1 & \delta E_1 &:= 0.01 \cdot E_1 & \delta \alpha_1 &:= 0.01 \cdot \alpha_1 & \delta \Delta T &:= 0.01 \cdot \Delta T \\ \delta t_2 &:= 0.01 \cdot t_2 & \delta E_2 &:= 0.01 \cdot E_2 & \delta \alpha_2 &:= 0.01 \cdot \alpha_2 \end{aligned}$$

$$\rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) := \frac{(t_1 + t_2) \cdot \left[3 \left(1 + \frac{t_1}{t_2} \right)^2 + \left(1 + \frac{t_1}{t_2} \cdot \frac{E_1}{E_2} \right) \cdot \left[\left(\frac{t_1}{t_2} \right)^2 + \frac{1}{\frac{t_1}{t_2} \cdot \frac{E_1}{E_2}} \right] \right]}{6 \cdot \left(1 + \frac{t_1}{t_2} \right)^2 \cdot (\alpha_2 - \alpha_1) \cdot \Delta T}$$

$$\delta \rho := \sqrt{\left(\frac{d}{dt_1} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta t_1 \right)^2 + \left(\frac{d}{dt_2} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta t_2 \right)^2 + \left(\frac{d}{dE_1} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta E_1 \right)^2 \dots} \\ + \left(\frac{d}{dE_2} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta E_2 \right)^2 + \left(\frac{d}{d\alpha_1} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta \alpha_1 \right)^2 + \left(\frac{d}{d\alpha_2} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta \alpha_2 \right)^2 \dots \\ + \left(\frac{d}{d\Delta T} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta \Delta T \right)^2$$

$$\% \delta t_1 := \frac{\left(\frac{d}{dt_1} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta t_1 \right)^2}{\delta \rho^2} \cdot 100 = 8.607$$

$$\% \delta t_2 := \frac{\left(\frac{d}{dt_2} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta t_2 \right)^2}{\delta \rho^2} \cdot 100 = 45.995$$

$$\% \delta E_1 := \frac{\left(\frac{d}{dE_1} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta E_1 \right)^2}{\delta \rho^2} \cdot 100 = 6.954$$

$$\% \delta E_2 := \frac{\left(\frac{d}{dE_2} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta E_2 \right)^2}{\delta \rho^2} \cdot 100 = 6.954$$

$$\% \delta \alpha_1 := \frac{\left(\frac{d}{d\alpha_1} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta \alpha_1 \right)^2}{\delta \rho^2} \cdot 100 = 16.629$$

$$\% \delta \alpha_2 := \frac{\left(\frac{d}{d\alpha_2} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta \alpha_2 \right)^2}{\delta \rho^2} \cdot 100 = 0.053$$

$$\% \delta \Delta T := \frac{\left(\frac{d}{d\Delta T} \rho(t_1, t_2, E_1, E_2, \alpha_1, \alpha_2, \Delta T) \cdot \delta \Delta T \right)^2}{\delta \rho^2} \cdot 100 = 14.808$$

$$\% \delta t_1 + \% \delta t_2 + \% \delta E_1 + \% \delta E_2 + \% \delta \alpha_1 + \% \delta \alpha_2 + \% \delta \Delta T = 100$$

Appendix C: Matlab programs

This appendix contains all the programs with corresponding flowcharts that we created and use for this project. The programs were all written in Matlab version 7.6.0.324(R2008a).

Beam deflection reading algorithm

This algorithm is aimed at reading the surface of the beams at several temperatures and determines their sensitivity to changes in temperatures.

Flowchart

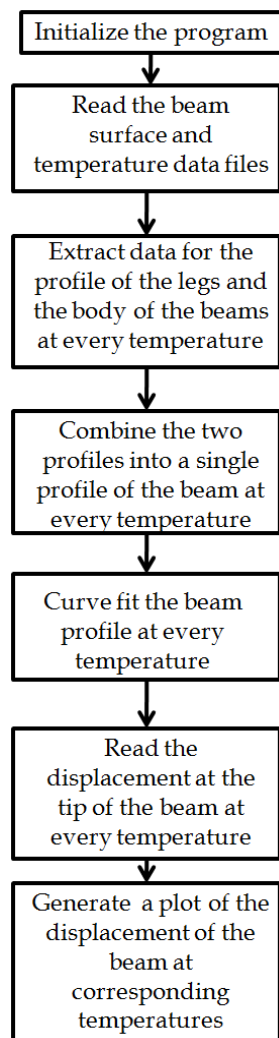


Fig. 40. Beam deflection reading algorithm flowchart.

Code of file Beam deflections 040309 1613.m

```
%% Initialization variables
clear all; clc;close all;

beam_lengh_um = 136.5;
beam_lengh_pixels = 111;
um_per_pixel=beam_lengh_um/beam_lengh_pixels;

%number of data files containing samples of data
number_of_samples =10;

%read first file to determine size

sample_location1      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 1\img01.xlsx';
sample_location2      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 2\img1.xlsx';
sample_location3      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 032509 3\img1.xlsx';
sample_location4      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 033009 4\img1.xlsx';
sample_location5      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 040309 5\img1.xlsx';
sample_location6      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 040309 6\img1.xlsx';

sample_location = sample_location5;

sample1 = xlsread(sample_location);
[rows_data,cols_data] = size(sample1);

%location of the data files
data_location1      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 1\img0';
data_location2      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 2\img';
data_location3      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 032509 3\img';
data_location4      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 033009 4\img';
data_location5      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 040309 5\img';
data_location6      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 040309 6\img';

data_location = data_location5;

%location of the temperature data file
temp_data_location2      =      'C:\ALL\School\MQP\Beams      sensitivity\Deflection
readings\Data 2\temperature_readings.xlsx';
```

```

temp_data_location3    =    'C:\ALL\School\MQP\Beams    sensitivity\Deflection
readings\Data 032509 3\temperature_readings.xlsx';
temp_data_location4    =    'C:\ALL\School\MQP\Beams    sensitivity\Deflection
readings\Data 033009 4\temperature_readings.xlsx';
temp_data_location5    =    'C:\ALL\School\MQP\Beams    sensitivity\Deflection
readings\Data 040309 5\temperature_readings.xlsx';
temp_data_location6    =    'C:\ALL\School\MQP\Beams    sensitivity\Deflection
readings\Data 040309 6\temperature_readings.xlsx';

temp_data_location = temp_data_location5;
%read temperature of samples in ohms
temperature_Kohms = xlsread(temp_data_location);
%convert to degrees C from Kohms using this equaiton
%0.000000006*x^6 - 0.000002*x^5 + 0.0002*x^4 - 0.0108*x^3 + 0.3572*x^2 -
7.1184*x + 69.736
f1 = inline('0.000000006*x.^6 - 0.000002*x.^5 + 0.0002*x.^4 - 0.0108*x.^3 +
0.3572*x.^2 - 7.1184*x + 69.736');
f2 = inline('x');
f = f2;
temperature_deg = f(temperature_Kohms);

%functuion to handle the desired naming scheme of the files
getName = @(num)IntToStr1(num,data_location);

%matrix to store all the data
beams_data_1 = zeros(rows_data,number_of_samples);
beams_data_2 = zeros(rows_data,number_of_samples);
beams_data_3 = zeros(rows_data,number_of_samples);
beams_data_mean = zeros(rows_data,number_of_samples);

legs_data_1 = zeros(rows_data,number_of_samples);
legs_data_2 = zeros(rows_data,number_of_samples);
legs_data_3 = zeros(rows_data,number_of_samples);
legs_data_mean = zeros(rows_data,number_of_samples);

current_sample = zeros(rows_data,cols_data);

%% read data
for i = 1:1:number_of_samples
    %generate new location name
    current_location = getName(i);
    %read file
    current_sample = xlsread(current_location);
    %extract beams displacement information
    beams_data_1(:,i) = current_sample(:,2);
    beams_data_2(:,i) = current_sample(:,3);
    %beams_data_3(:,i) = current_sample(:,4);
    %generate beams average displacement information
    beams_data_mean(:,i)    =    (beams_data_1(:,i)+    beams_data_2(:,i))/2;
    %+beams_data_3(:,i))/3;

    %extract legs displacement information
    legs_data_1(:,i) = current_sample(:,1);

```



```

    legs_data_2(:,i) = current_sample(:,4);
    %legs_data_3(:,i) = current_sample(:,6);
    %generate legs average displacement information
    legs_data_mean(:,i) = (legs_data_1(:,i)+ legs_data_2(:,i))/2; % +
    legs_data_3(:,i))/3;

end

%data = cat(2,beams_data,legs_data);

%% read beam locations

beam_locations_data2 = 'C:\ALL\School\MQP\Beams sensitivity\Deflection
readings\Data 2\Beam boarders.xlsx';
beam_locations_data3 = 'C:\ALL\School\MQP\Beams sensitivity\Deflection
readings\Data 032509 3\Beam boarders.xlsx';
beam_locations_data4 = 'C:\ALL\School\MQP\Beams sensitivity\Deflection
readings\Data 033009 4\Beam boarders.xlsx';
beam_locations_data5 = 'C:\ALL\School\MQP\Beams sensitivity\Deflection
readings\Data 040309 5\Beam boarders.xlsx';
beam_locations_data6 = 'C:\ALL\School\MQP\Beams sensitivity\Deflection
readings\Data 040309 6\Beam boarders.xlsx';

beam_locations_data = beam_locations_data5;

%read temperature of samples in ohms
beam_locations = xlsread(beam_locations_data);

%% extract the particular beam data

close all;
%parameters for filtering the data of only one beam
beam_location = 1;
current_beam_location = beam_locations(beam_location,:);

start_of_beam = current_beam_location(1,1);
end_of_beam = current_beam_location(1,2);
beam_range = end_of_beam - start_of_beam +1;

start_of_leg = end_of_beam;
end_of_leg = current_beam_location(1,3);
leg_range = end_of_leg - start_of_leg +1;

total_range = beam_range+leg_range;
total_beam_range = 1:1:total_range;
total_beam_range = total_beam_range';

range_mid = int32((beam_range+leg_range)/2);

beam_data = zeros(beam_range,number_of_samples);
leg_data = zeros(leg_range,number_of_samples);

for i = 1:1:number_of_samples

```

```

    beam_data(:,i) = beams_data_mean(start_of_beam:end_of_beam,i);
    leg_data(:,i) = legs_data_mean(start_of_leg:end_of_leg,i);
end

data_beam_raw = cat(1,beam_data,leg_data);

%% line up all of the beam instances
%*****

data_beam = data_beam_raw;
%read position of the beginning of each beam instance
offset = data_beam((beam_range+leg_range),:);
%offset data
for i = 1:1:number_of_samples
    data_beam(:,i) = data_beam(:,i) - offset(1,i);
end

%read the beam tip positions
beam_tip_disp = data_beam(1,:);

%reoriend the beams so that the base is on the left
data_beam = flipud(data_beam);

%% Fit data and calculate displacement
%*****

%generate x range for the lenght of the beam
x_range_pixels = total_beam_range;
x_range_um = (x_range_pixels-1)*um_per_pixel;

%generate x range for the degees
x_range_deg = temperature_deg;
x_range_deg_min = min(x_range_deg(:,1));
x_range_deg_delta = x_range_deg(:,1) - x_range_deg_min;

beam_tip_fit = zeros(number_of_samples,1);
beam_fit = zeros(total_range,number_of_samples);

%degree of polynomial fit
poly_n = 2;
%position for calculation of displacement
beam_tip_x = max(x_range_um);
% Matrix to store the coefficients for the polynomial fit of the
%deflection of the surface of each beam at each temperature.
all_poly = zeros(number_of_samples,poly_n+1);

for i = 1:1:number_of_samples

```

```

current_poly = polyfit(x_range_um,data_beam(:,i),poly_n);
all_poly(i,:) = current_poly(1,:);
beam_tip_fit(i,1) = polyval(current_poly,beam_tip_x);
for j = 1:1:total_range
    current_x = x_range_um(j,1);
    beam_fit(j,i) = polyval(current_poly,current_x);
end
end

%line up all of fitted beams*****
%read position of the beginning of each beam instance
offset = beam_fit(1,:);
%offset data
for i = 1:1:number_of_samples
    beam_fit(:,i) = beam_fit(:,i) - offset(1,i);
end

%% fit final data
%*****
lin_p = polyfit(x_range_deg_delta,beam_tip_fit,1);
beam_tip_fit_linear = polyval(lin_p,x_range_deg_delta);
fprintf(' \n');
fprintf('\n Beam number: %g \n', beam_location);
fprintf('Fitted data coefficients: \n');
fprintf(' %g \n', lin_p);

%% offset fit and refit data
%*****
beam_tip_fit_linear_max = max(beam_tip_fit_linear);
beam_tip_fit_offsetted = beam_tip_fit_linear - beam_tip_fit_linear_max;

lin_p_offsetted = polyfit(x_range_deg_delta,beam_tip_fit_offsetted,1);
beam_tip_fit_linear_offsetted = polyval(lin_p_offsetted,x_range_deg_delta);
fprintf(' \n');
fprintf('\n Beam number: %g \n', beam_location);
fprintf('Refitted data coefficients: \n');
fprintf(' %g \n', lin_p_offsetted);

%% Computational results

computational_data_location1 = 'C:\ALL\School\MQP\Beams
sensitivity\Deflection readings\Computational data\Computational_data.xlsx';
computational_data_location = computational_data_location1;

%read temperature of samples in ohms
computational_data = xlsread(computational_data_location);

%offset between the beam beginning location of the computational and the
%experimental data [um]
computational_offset = (beam_lengh_um - max(x_range_um))/2;
%find the max of the data
x_range_um_computational_max = max(x_range_um) + computational_offset;
%get the size
computational_data_size = size(computational_data,1);

```

```

%generate a x range with the give size
step_x_comp =x_range_um_computational_max /computational_data_size;
x_range_um_computational = 0:step_x_comp:x_range_um_computational_max;

%fit equation for the thermal sensitivity of the beam with respect to its
length
computational_data_poly =
polyfit(computational_data(:,1),computational_data(:,2),2);

computational_data_defl_per_deg_offset = polyval(computational_data_poly,
computational_offset);
computational_data_defl_per_deg = polyval(computational_data_poly,
x_range_um_computational_max);
computational_data_defl_per_deg = computational_data_defl_per_deg +
computational_data_defl_per_deg_offset;
fprintf(' \n');
fprintf('Computational data slope: \n');
fprintf(' %g \n', computational_data_defl_per_deg);

%beam_tip_disp_computational_func =
computational_data_defl_per_deg*x_range_deg_delta; % inline('-0.1059*x-
1*10^(-7)');
beam_tip_disp_computational =
computational_data_defl_per_deg*x_range_deg_delta;

%% plot data
%*****
close all;
sample_set =4;
range = start_of_beam:1:end_of_leg;
x_dummy = 1:1:rows_data;

figure(1)
plot(x_dummy,beams_data_mean(:,sample_set),'r');
str1(1) = {'\bullet\leftarrow Beginning of beam data'};
text1_x_pos = start_of_beam;
text1_y_pos = beams_data_mean(text1_x_pos,sample_set);
text(text1_x_pos,text1_y_pos,str1);
str2(1) = {'\bullet\leftarrow End of beam data'};
text2_x_pos = end_of_beam;
text2_y_pos = beams_data_mean(text2_x_pos,sample_set);
text(text2_x_pos,text2_y_pos,str2);
str3(1) = {'\bullet\leftarrow End of leg data'};
text3_x_pos = end_of_leg;
text3_y_pos = legs_data_mean(text3_x_pos,sample_set);
text(text3_x_pos,text3_y_pos,str3);

hold on
plot(x_dummy,legs_data_mean(:,sample_set),'g');
xlabel('X position along the sample [pixels]');
ylabel('Y position [um]');
title('raw data - full');

h = legend('Beams data','Legs data',1);

```

```

set(h, 'Interpreter', 'none')
grid on
axis tight

figure(2)
plot(x_dummy(range),beams_data_mean(range,sample_set), 'r');
hold on
plot(x_dummy(range),legs_data_mean(range,sample_set), 'g');
grid on
xlabel('X position along the sample [pixels]');
ylabel('Y position [um]');
title('beam and leg raw data');

h = legend('Beam data','Leg data',1);
set(h, 'Interpreter', 'none')
axis tight

figure(3)
plot(x_range_um,flipud(data_beam_raw));
grid on
xlabel('X position along the sample [pixels]');
ylabel('Y position [um]');
title('Curvature of selected beam at each temperature - raw data');
axis tight

figure(4);
plot(x_range_um,data_beam);
xlabel('X position along the beam [um]');
ylabel('Displacement [um]');
title('Curvature of selected beam at each temperature - offsetted raw data
');
grid on
axis tight

figure(5);
plot(x_range_um,beam_fit);
xlabel('X position along the beam [um]');
ylabel('Displacement [um]');
title('Curvature of selected beam at each temperature - fitted data ');
grid on
axis tight

beam_instance = 2;

figure(6);
plot(x_range_um,beam_fit(:,beam_instance),
x_range_um,data_beam(:,beam_instance));
xlabel('X position along the beam [um]');
ylabel('Displacement [um]');
title('Curvature of selected beam at each temperature - fitted and offsetted
raw data ');
grid on

```

```

axis tight

figure(7);
plot(x_range_deg_delta,beam_tip_disp,x_range_deg_delta,beam_tip_fit,'g',x_range_deg_delta,beam_tip_fit_linear,'r');
xlabel('Change in temp [Deg C]') ;
ylabel('Displacement [um]');
title('Displacement vs change in temperature');
h = legend('beam tip disp raw data','beam tip disp fitted data','beam tip disp fitted data linear approximation',1);
set(h,'Interpreter','none')
grid on
axis tight

figure(8);
plot(x_range_deg_delta,beam_tip_disp_computational,'-+r',...
     x_range_deg_delta,beam_tip_fit_linear_offsetted);
xlabel('Change in temp [Deg C]') ;
ylabel('Displacement [um]');
title('Displacement vs change in temperature - comparison of methods');
h = legend('computational method','analytical method','experimental method',1);
set(h,'Interpreter','none')
grid on
axis tight

```

Masking algorithm

This algorithm is aimed at generating a mask that makes visible only the beams and certain parts of the substrate, thus reducing the noise in the data.

Flowchart

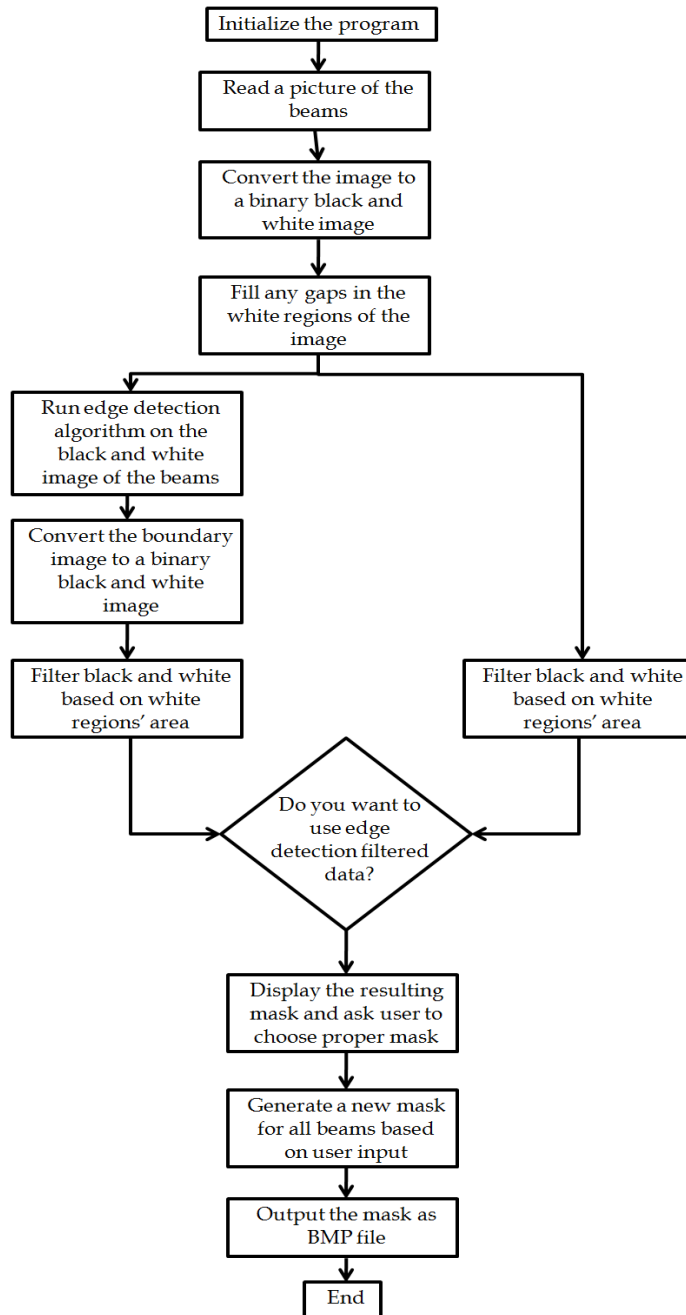


Fig. 41. Masking algorithm flowchart

Code of file Masking 022409 1130.m

```
%% Initialization variables
clear all; clc;close all;

%location of the files
location1 = 'C:\ALL\School\MQP\Beams          sensitivity\Edge
detection\img0004_1.tif';
location2 = 'H:\MQP\Masking\img0004_1.tif';
location3 = 'C:\sample_dot_grid.bmp';

location=location1;

raw_data = imread(location); %reads data
data_bits = raw_data(:,:,1);
[rows_data,cols_data,d] = size(raw_data); %reads size of images

H = fspecial('gaussian');
I_sharp = imfilter(data_bits,H,'replicate');
%I_inv = inv(data_bits);
figure(1)
subplot(1,2,1)
imshow(data_bits);
title('original');
subplot(1,2,2)
imshow(I_sharp);
title('original sharp');

%% Detect edges on raw image

%
% I_sobel_raw = edge(data_bits,'sobel');
% I_sobel_sharp_raw = edge(I_sharp,'sobel');
% I_canny_raw = edge(data_bits,'canny');
% I_canny_sharp_raw = edge(I_sharp,'canny');
%
% figure(2)
% subplot(2,2,1)
% imshow(I_sobel_raw)
% title('sobel edge detection raw');
% subplot(2,2,2)
% imshow(I_sobel_sharp_raw)
% title('sobel sharp edge detection raw');
% subplot(2,2,3)
% imshow(I_canny_raw)
% title('canny edge detection raw');
% subplot(2,2,4)
% imshow(I_canny_sharp_raw)
```



```

% title('canny sharp edge detection raw');

%% covert to BW
I_bw_level = graythresh(data_bits);

I_bw = im2bw(data_bits,0.38);
I_bw_sharp = im2bw(I_sharp,I_bw_level);

figure(3)
subplot(1,2,1)
imshow(I_bw)
title('Black and white image');
subplot(1,2,2)
imshow(I_bw_sharp)
title('Black and white image  sharpened');

I3 = I_bw_sharp;

%% fill bw image
I_filled = imfill(I3,'holes');

%H = fspecial('unsharp');
%I_filled_bw_sharp = imfilter(I_filled,H,'replicate');

figure(4)
%subplot(1,2,1)
imshow(I_filled);
title('BW filled');
%subplot(1,2,2)
%imshow(I_filled_bw_sharp);
%title('BW filled - sharpened');

I3_filled = I_filled;
%% Detect edges on BW filled image

I3_uint8_filled = cast(I3_filled,'uint8');
%I3_uint8_filled_sharp = cast(I_filled_bw_sharp,'uint8');

I_filled_edges = edge(I3_uint8_filled,'canny');
%I_filled_BW_sharp = edge(I3_uint8_filled_sharp,'canny');

figure(5)
%subplot(1,2,1)
imshow(I_filled_edges)
title('canny edge detection on BW filled');
%subplot(1,2,2)
%imshow(I_filled_BW_sharp)
%title('canny edge detection on BW filled - sharpened');

%% boundary detection
[B1,L1] = bwboundaries(I_filled_edges,4,'noholes'); %data_bits,I_filled_BW
%s_center = regionprops(L1, 'centroid');

```

```

%boundary_centroids = cat(1, s_center.Centroid);

figure(6)
imshow(data_bits)
title('boundaries of possible elements');
hold on
for k = 1:length(B1)
    boundary = B1{k};
    plot(boundary(:,2), boundary(:,1), 'g', 'LineWidth', 1)
    hold on
    %drawnow;
end
%plot(boundary_centroids(:,1), boundary_centroids(:,2), 'b+')

%% convert boundary image to BW image
%fill the holes in the image
L1_filled = imfill(L1, 'holes');
%determine treshhold for BW conversion
I_filled_edges_bw_level = graythresh(L1_filled);
% convert to BW image
L1_filled_bw = im2bw(L1_filled, I_filled_edges_bw_level);

figure(7)
subplot(1,2,1)
imshow(L1_filled);
title('boundaries of possible elements');

subplot(1,2,2)
imshow(L1_filled_bw);
title('BW boundaries of possible elements');

%% find one boundary

% X_pos = 650;
% Y_pos = 650;
%
% I3_filled_invr = ~I3_filled;
% boundary_1 = bwtraceboundary(I3_filled, [Y_pos, X_pos], 'E');
% %boundary_1_sharp = bwtraceboundary(I_filled_bw_sharp, [Y_pos, X_pos], 'N');
%
% figure(7)
% %subplot(1,2,1)
% imshow(I3_filled)
% hold on;
% plot(boundary_1(:,2), boundary_1(:,1), 'g', 'LineWidth', 1);
% drawnow;
% title('boundary of BW image');
%
%
% % subplot(1,2,2)
% % imshow(I_filled_BW_sharp)
% % hold on;
% % plot(boundary_1_sharp(:,2), boundary_1_sharp(:,1), 'g', 'LineWidth', 1);

```

```

%% title('boundary of sharpened BW image');

%% filter

%**** filter BW data w/o edge detection*****
L_unfiltered = bwlabel(I3_filled,4);

max_area = 4700;
min_area = 1800;

s_area = regionprops(L_unfiltered, 'Area');
check = ([s_area.Area]> min_area)&([s_area.Area]< max_area);
idx = find(check);
I3_filled_filtered = ismember(L_unfiltered,idx);

%**** filter BW data w/ edge detection*****

L_unfiltered_edge = bwlabel(L1_filled_bw,4);

max_area_edge = 4700;
min_area_edge = 1980;

s_area_edge = regionprops(L_unfiltered_edge, 'Area');
check_edge = ([s_area_edge.Area]> min_area_edge)&...
    ([s_area_edge.Area]< max_area_edge);
idx_edge = find(check_edge);
I3_filled_filtered_edge = ismember(L_unfiltered_edge,idx_edge);

figure(8);
subplot(2,1,1)
imagesc(I3_filled_filtered)
title('filtered BW regions');

subplot(2,1,2)
imagesc(I3_filled_filtered_edge)
title('filtered BW regions with edge detection');

filtering_ok = input('Do you want to use edge detection filtered data? Y/N
[Y]: ','s');
if isempty(filtering_ok)
    filtering_ok = 'Y';
end

if (filtering_ok == 'Y')
    I3_filled_filtered = I3_filled_filtered_edge;
end
%% display resulting masking

```

```

L_filtered = bwlabeln(I3_filled_filtered,4);
s_center = regionprops(L_filtered, 'centroid');
centroids = cat(1, s_center.Centroid);

data_masked = data_bits;

for i = 1:1:rows_data
    for j = 1:1:cols_data
        if (L_filtered(i,j)==0)
            data_masked(i,j) = 0;
        end
    end
end

figure(9)
imshow(data_masked);
hold on
title('Resulting masking');
plot(centroids(:,1), centroids(:,2), 'g+')

%% region selection
selection_ok = 'N';

figure(10)
imagesc(L_filtered);
hold on
title('Please select desired region by entering the index of the region');
plot(centroids(:,1), centroids(:,2), 'g+')

while (selection_ok == 'N')

Region_index = input('What is the index of the desired region: ');

L_desired = ismember(L_filtered,Region_index);

data_masked_desired = data_bits;
for i = 1:1:rows_data
    for j = 1:1:cols_data
        if (L_desired(i,j)==0)
            data_masked_desired(i,j) = 0;
        end
    end
end

figure(11)
imshow(data_masked_desired);
hold on
title('Desired mask');
plot(centroids(Region_index,1), centroids(Region_index,2), 'g+')

```

```

% *****Create full mask *****
Desired_object = bwlabeln(L_desired,4);
Stats_object = regionprops(Desired_object, 'centroid','Extrema');

center_col = Stats_object.Centroid(1,2);
center_row = Stats_object.Centroid(1,1);

trans_cols = 62;
trans_rows = 62;

border_width = 4;
border_lenght = 10;
border_buffer = 1;

T_mask_X = int32((Stats_object.Extrema(1,1)+ Stats_object.Extrema(2,1))/2);
T_mask_Y = int32((Stats_object.Extrema(1,2)+ Stats_object.Extrema(2,2))/2);

R_mask_X = int32((Stats_object.Extrema(3,1)+ Stats_object.Extrema(4,1))/2);
R_mask_Y = int32((Stats_object.Extrema(3,2)+ Stats_object.Extrema(4,2))/2);

B_mask_X = int32((Stats_object.Extrema(5,1)+ Stats_object.Extrema(6,1))/2);
B_mask_Y = int32((Stats_object.Extrema(5,2)+ Stats_object.Extrema(6,2))/2);

L_mask_X = int32((Stats_object.Extrema(7,1)+ Stats_object.Extrema(8,1))/2);
L_mask_Y = int32((Stats_object.Extrema(7,2)+ Stats_object.Extrema(8,2))/2);

mask_width = abs(B_mask_Y - T_mask_Y);
mask_lenght = abs(R_mask_X - L_mask_X);

inter_element_width = abs(trans_rows - mask_width);
rail_width = abs(inter_element_width - 2*border_width);

A_X = R_mask_X - border_buffer;
A_Y = T_mask_Y - border_width;

B_X = A_X - border_lenght;
B_Y = T_mask_Y + border_buffer;

C_X = L_mask_X - border_buffer;
C_Y = A_Y - rail_width;

D_X = B_X;
D_Y = C_Y;

E_X = A_X;
E_Y = C_Y - border_width - border_buffer;

Full_mask_element = L_desired;
%patch lower border
Full_mask_element(A_Y:B_Y, B_X:A_X) = 1;
%patch rail

```

```

Full_mask_element(C_Y:A_Y,C_X:A_X ) = 1;
%patch upper border
Full_mask_element(E_Y:D_Y, D_X:E_X) = 1;

data_masked_desired_full_element = data_bits;
for i = 1:1:rows_data
    for j = 1:1:cols_data
        if (Full_mask_element(i,j)==0)
            data_masked_desired_full_element(i,j) = 0;
        end
    end
end

figure(12)
imshow(data_masked_desired_full_element);
grid on
hold on
title('Desired mask full element');
plot(centroids(Region_index,1), centroids(Region_index,2), 'g+')

% ***** pattern mask *****

X_center = centroids(Region_index,1);
X_total = cols_data;
X_delta = trans_rows;

Y_center = centroids(Region_index,2);
Y_total = rows_data;
Y_delta = trans_cols;

iterations_up = floor(Y_center/Y_delta) - 1;
iterations_down = floor(abs(Y_total - Y_center)/Y_delta);
iterations_left = floor(X_center/X_delta);
iterations_right = floor(abs(X_total - X_center)/X_delta) - 1;

total = data_masked_desired_full_element;

%transformations
se_down = translate(strel(1), [trans_cols 0]);
se_up = translate(strel(1), [-trans_cols 0]);
se_right = translate(strel(1), [0 trans_rows]);
se_left = translate(strel(1), [0 -trans_rows]);

% Pattern up
J_prev = data_masked_desired_full_element;
for i = 1:1:iterations_up
    J_next = imdilate(J_prev,se_up);
    total = (J_next | total);
    J_prev = J_next;
end

```

```

% Pattern down
J_prev = data_masked_desired_full_element;
for i = 1:1:iterations_down
    J_next = imdilate(J_prev,se_down);
    total = (J_next | total);
    J_prev = J_next;
end

%Pattern left
J_prev = total;
total_col = total;
for i = 1:1:iterations_left
    J_next = imdilate(J_prev,se_left);
    total = (J_next | total);
    J_prev = J_next;
end

% Pattern right
J_prev = total_col;
for i = 1:1:iterations_right
    J_next = imdilate(J_prev,se_right);
    total = (J_next | total);
    J_prev = J_next;
end

% mix real image with mask
data_masked_final = data_bits;
for i = 1:1:rows_data
    for j = 1:1:cols_data
        if (total(i,j)==0)
            data_masked_final(i,j) = 0;
        end
    end
end

figure(13)
imshow(data_masked_final)
title('final result of masking');
hold on
plot(centroids(Region_index,1), centroids(Region_index,2), 'g+')

figure(14)
imshow(total)
title('final mask');
hold on
plot(centroids(Region_index,1), centroids(Region_index,2), 'g+')

```

```

selection_ok = input('Are you ok with your selection? Y/N [Y]: ', 's');
if isempty(selection_ok)
    selection_ok = 'Y';
end

end

%% output result

output_location_mask = 'C:\ALL\School\MQP\Beams sensitivity\Edge
detection\img0004_1_mask.tif';
imwrite(total,output_location_mask,'tif','WriteMode','overwrite');

%% apply mask
%location of the files
location_target1 = 'H:\MQP\Masking\img0006.tif';
location_target2 = 'C:\ALL\School\MQP\Beams sensitivity\Edge
detection\img0006.tif';

location_target=location_target2;

%read target image
raw_data_target = imread(location_target); %reads data
data_target = raw_data_target(:,:,1);

% mix real image with mask
data_target_masked = data_target;
for i = 1:1:rows_data
    for j = 1:1:cols_data
        if (total(i,j)==0)
            data_target_masked(i,j) = 0;
        end
    end
end

% save masked image
output_location_masked = 'C:\ALL\School\MQP\Beams sensitivity\Edge
detection\img0006_masked.tif';
imwrite(data_target_masked,output_location_masked,'tif',...
'WriteMode','overwrite');

```


Calibration algorithm

This algorithm is aimed at determining the voltage at which the piezo-actuator will produce displacement corresponding to $\pi/4$ phase shift in the reference beam.

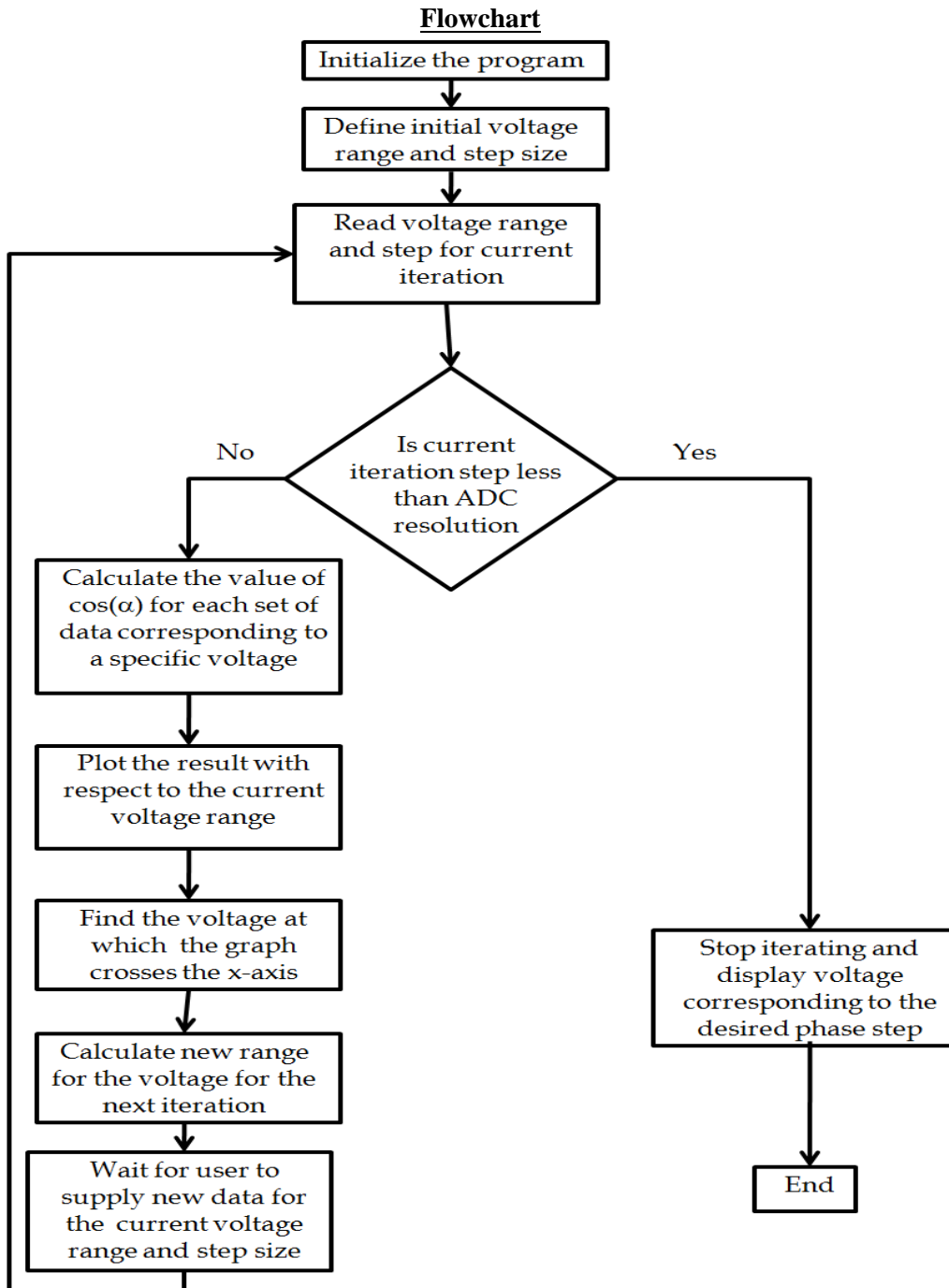


Fig. 42. Calibration algorithm flowchart.

Code of file Calibrate_042809.m

```
%% Initialization variables
clear all; clc;close all;
%Upper and lower border of voltage range
V_min_next = 3;
V_max_next = 4;
%voltage step
min_step_next =0.05;
%bit range of the data
bit_range = 0:1:255;
%Number of frames per sample
frames = 5;

%location of the files
location1 ='C:\ALL\School\MQP\PZ mapping\Accuracy test 1 020609\data 020609
1\IMG_0001_';
location2 ='C:\ALL\School\MQP\PZ mapping\Accuracy test 1 020609\data 020609
2\IMG_0001_';
location3 ='C:\ALL\School\MQP\PZ mapping\Accuracy test 1 020609\data 020609
3\IMG_0001_';
location4 ='C:\ALL\School\MQP\PZ mapping\Accuracy test 1 020609\data 020609
4\IMG_0001_';
location5 ='C:\ALL\School\MQP\PZ mapping\Accuracy test 1 020609\data 020609
5\IMG_0001_';
location6 ='C:\ALL\School\MQP\PZ mapping\Accuracy test 1 020609\data 020609
6\IMG_0001_';
location7 ='C:\ALL\School\MQP\PZ mapping\Accuracy test 1 020609\data 020609
7\IMG_0001_';

location=location1;

%minimum for the absolute value of the denominator in the cosine formula
cos_min = 0.01;

%iterator for the names of the files
name=0;
%max number of digits for thenaming of the files
max_name = 4;

raw_line_voltage_result = 0;
raw_mean_voltage_result = 0;

%overall parameters for the iterations
min_step_absolute = 0.05;
step_change_factor = 2;
lower_limit = 0.9;
upper_limit=1.1;
continue_flag = 1;
reply = 'N';

%% Iterate
```

```

while(reply ~= 'Y')
    reply = input('Do you want to start calibration? Y/N [Y]: ', 's');

    %default answer is Y
    if isempty(reply)
        reply = 'Y';
    end
    %stop if N
    if (reply == 'N')
        continue_flag=0;
        fprintf('\n Calibration algorithm stopped! \n');

        return;
    end
end

while (continue_flag==1)
    fprintf('\n Calculating..... \n');
    Iterate_042809_1;

    if (min_step_next<= min_step_absolute)
        fprintf('\n The algorithm has converged to a final value \n');
        fprintf(' Line: %g \n', raw_line_voltage_result);
        fprintf(' Line_1: %g \n', raw_line_voltage_result_1);
        fprintf(' Mean: %g \n', raw_mean_voltage_result);
        fprintf(' Mean_1:%g \n', raw_mean_voltage_result_1);
        continue_flag=0;
        return;
    else
        fprintf('\n Current iteration finished\n');
        fprintf('\n Current results: \n');
        fprintf(' Line: %g \n', raw_line_voltage_result);
        fprintf(' Line_1: %g \n', raw_line_voltage_result_1);
        fprintf(' Mean: %g \n', raw_mean_voltage_result);
        fprintf(' Mean_1: %g \n', raw_mean_voltage_result_1);

        reply = input('Do you want to continue calibration? Y/N [Y]: ', 's');
        %default answer is Y
        if isempty(reply)
            reply = 'Y';
        end

        %stop if N
        if (reply == 'N')
            continue_flag=0;
            fprintf('\n Calibration algorithm stopped! \n');
            return;
        end

        %continue if Y
        if (reply == 'Y')
            continue_flag=1;
            fprintf('\n Use the following parameters for next iteration: \n');
            fprintf(' Porn Vmin: %g \n', V_min_next);
            fprintf(' Vmax: %g \n', V_max_next);
        end
    end
end

```

```

fprintf(' Step: %g \n', min_step_next);

reply = input('Ready to continue? Y/N [Y]: ', 's');
%default answer is Y
if isempty(reply)
    reply = 'Y';
end
%stop if N
if (reply == 'N')
    continue_flag=0;
    fprintf('\n Calibration algorithm stopped! \n');
    return;
end

end
end

end

```

Code of file Iterate_042809_1.m

```

%% Initialization variables
% clear all; clc;close all;

%Upper and lower border of voltage range
Down_range = V_min_next;
Up_range = V_max_next;
%Voltage range
range = Up_range - Down_range;
%voltage step
Voltage_step =min_step_next;
%voltage samples
Voltage = Down_range:Voltage_step:Up_range;

% bit_range = 0:1:255;
%Number of sets of samples
sets = round(range/Voltage_step+1);

%fucntion for creating names of type IMGnumberOfPic
num2name = @(num) IntToStr(num,max_name,location);

%fucntion for creating names of type test0001_voltageSet_stepNumber
num2name2 = @(num) IntToStr2(num,frames, location, Voltage_step,Down_range);

%functuion to handle the desired naming scheme of the files
getName = @(num) num2name2(num);

rgbToGray = @(rgb_pic) RGBtoGray(rgb_pic);

```

```

%str iterator for the names of the files
str_name = getName(name);

%reads the first image to measure the size of the image so that it can
%preallocate space. It assumes that all the images are the same.
raw_pic = imread(str_name); %reads first image
[rows_pix,cols_pix,depth] = size(raw_pic); %reads size of images

%viewing area parameters
Col_start = int32(0.45*cols_pix);
Col_end = int32(0.55*cols_pix);
Col_range = Col_end-Col_start+1;

Row_start = int32(0.05*rows_pix);
Row_end = int32(0.95*rows_pix);
Row_range = Row_end-Row_start+1;

%% Read data

%allocate space for a matrix to hold the intensities and phase shifts
%of the several measurements
I=zeros(Row_range,Col_range,frames); %temporary stores intensities of a
sample of frames
I_data=zeros(Row_range,Col_range,frames, sets);

I_1=zeros(Row_range,Col_range);
I_2=zeros(Row_range,Col_range);
I_3=zeros(Row_range,Col_range);
I_4=zeros(Row_range,Col_range);
I_5=zeros(Row_range,Col_range);

cos_alpha_raw = zeros(Row_range,Col_range);
cos_alpha_data_raw = zeros(Row_range,Col_range,sets);

for set = 1:1:sets
    %reads and converts from RGB to grayscale
    for frame = 1:1:frames

        raw_pic = imread(str_name); %read next file
        raw_pic = cast(raw_pic, 'double');%cast to double
        pic = rgbToGray(raw_pic); %convert to gray scale image
        %pic = rgb2gray(raw_pic);

        %read intensities of each frame for the particular sample
        I(:, :, frame)=pic(Row_start:Row_end, Col_start:Col_end);

        name = name+1; %iterate name of file
        str_name = getName(name); %generate the new name

    end;

    I_data(:, :, :, set) = I(:, :, :, :);

```

```

    %split intensity data for the particular sample into data for individual
frames

    I_1=I(:,:,1);
    I_2=I(:,:,2);
    I_3=I(:,:,3);
    I_4=I(:,:,4);
    I_5=I(:,:,5);

    cos_alpha_raw = (I_5-I_1)./(2*(I_4-I_2));
    cos_alpha_data_raw(:,:,set) = cos_alpha_raw;

end;
%% Preallocate space
Voltage=Voltage';

desired_row = int32(0.5*Row_range);
desired_col = int32(0.5*Col_range);

%*****final results data*****
alpha_raw_mean = zeros(Row_range,sets);
alpha_raw_mean_avg = zeros(sets,1);
alpha_raw_line = zeros(Row_range,sets);
alpha_raw_line_avg = zeros(sets,1);

% *****All rows mean data*****
I_1_mean= zeros(Row_range,1);
I_2_mean= zeros(Row_range,1);
I_3_mean= zeros(Row_range,1);
I_4_mean= zeros(Row_range,1);
I_5_mean= zeros(Row_range,1);
%I_denom_mean = zeros(sets, Row_range);
cos_alpha_raw_mean = zeros(Row_range,sets);
cos_alpha_raw_mean_avg = zeros(sets,1);

% *****Row data*****
I_1_line = zeros(Row_range,1);
I_2_line = zeros(Row_range,1);
I_3_line = zeros(Row_range,1);
I_4_line = zeros(Row_range,1);
I_5_line = zeros(Row_range,1);
%I_denom_line=zeros(sets, Row_range);
cos_alpha_raw_line = zeros(Row_range,sets);
cos_alpha_raw_line_avg = zeros(sets,1);

%% Calculate

for current_set = 1:1:sets
    % *****All cols mean data*****
for current_row = 1:1:Row_range
    I_1_mean(current_row,1)=mean(I_data(current_row,:,1,current_set));
    I_2_mean(current_row,1)=mean(I_data(current_row,:,2,current_set));
    I_3_mean(current_row,1)=mean(I_data(current_row,:,3,current_set));

```

```

I_4_mean(current_row,1)=mean(I_data(current_row,:,4,current_set));
I_5_mean(current_row,1)=mean(I_data(current_row,:,5,current_set));
end

%I_denom_mean(current_set,:) = I_4_mean(1,:) - I_2_mean(1,:);
cos_alpha_raw_mean(:,current_set) = (I_5_mean(:,1)-
I_1_mean(:,1))./(2*(I_4_mean(:,1)-I_2_mean(:,1)));

Counter = 0;
Sum = 0;
Limit = 0;
for i=1:Row_range
    Limit = 1 - abs(cos_alpha_raw_mean(i,current_set));
    if (Limit>cos_min)
        %alpha_raw_mean(current_set,i) =
acos(cos_alpha_raw_mean(current_set,i));
        Sum = Sum + cos_alpha_raw_mean(i,current_set);
        Counter = Counter+1;
    end
end
%alpha_raw_mean_avg(current_set,1) = Sum/Counter;
cos_alpha_raw_mean_avg(current_set,1) = Sum/Counter;
Counter = 0;
Sum = 0;

% *****Col data*****
I_1_line(:,1)=I_data(:,desired_col,1,current_set);
I_2_line(:,1)=I_data(:,desired_col,2,current_set);
I_3_line(:,1)=I_data(:,desired_col,3,current_set);
I_4_line(:,1)=I_data(:,desired_col,4,current_set);
I_5_line(:,1)=I_data(:,desired_col,5,current_set);

%I_denom_line(current_set,:) = I_4_line(1,:) - I_2_line(1,:);
cos_alpha_raw_line(:,current_set) = (I_5_line(:,1)-
I_1_line(:,1))./(2*(I_4_line(:,1)-I_2_line(:,1)));

Counter = 0;
Sum = 0;
Limit = 0;
for i=1:Row_range
    Limit = 1 - abs(cos_alpha_raw_line(i,current_set));
    if (Limit>cos_min)
        %alpha_line_mean(current_set,i) =
acos(cos_alpha_line_mean(current_set,i));
        Sum = Sum + cos_alpha_raw_line(i,current_set);
        Counter = Counter+1;
    end
end
%alpha_raw_line_avg(current_set,1) = Sum/Counter;
cos_alpha_raw_line_avg(current_set,1) = Sum/Counter;
Counter = 0;
Sum = 0;

end

```

```

%% Find voltage

%*****line*****

%Calculate the absolute values
cos_alpha_raw_line_avg_abs = abs(cos_alpha_raw_line_avg);
%Extract the point closest to zero as well as its index in the array
[cos_min,cos_index] = min(cos_alpha_raw_line_avg_abs);
%look at the real value of the element with that index
cos_zero_approx = cos_alpha_raw_line_avg(cos_index,1);
%raw_line_voltage_result = Voltage(cos_alpha_raw_line_avg_index,1);

%if it is above zero read the next element(below zero) and interpolate
if (cos_zero_approx>=0)
    %check if the next element exists
    if (cos_index+1>sets)%if not interpolate with the previous element
        left_cos = cos_alpha_raw_line_avg(cos_index-1,1);
        left_voltage = Voltage(cos_index-1,1);

        right_cos = cos_alpha_raw_line_avg(cos_index,1);
        right_voltage = Voltage(cos_index,1);

    else%if yes interpolate with the next element
        left_cos = cos_alpha_raw_line_avg(cos_index,1);
        left_voltage = Voltage(cos_index,1);

        right_cos = cos_alpha_raw_line_avg(cos_index+1,1);
        right_voltage = Voltage(cos_index+1,1);
    end
end

else
    %check if the previous element exists
    if (cos_index-1<0)%if not interpolate with the next element
        left_cos = cos_alpha_raw_line_avg(cos_index,1);
        left_voltage = Voltage(cos_index,1);

        right_cos = cos_alpha_raw_line_avg(cos_index+1,1);
        right_voltage = Voltage(cos_index+1,1);

    else%if yes interpolate with the previous element
        left_cos = cos_alpha_raw_line_avg(cos_index-1,1);
        left_voltage = Voltage(cos_index-1,1);

        right_cos = cos_alpha_raw_line_avg(cos_index,1);
        right_voltage = Voltage(cos_index,1);
    end
end

end
raw_line_voltage_result = left_voltage + (right_voltage-left_voltage)*(0-
left_cos)/(right_cos-left_cos);

%*****mean*****

```



```

%Calculate the absolute values
cos_alpha_raw_mean_avg_abs = abs(cos_alpha_raw_mean_avg);
%Extract the point closest to zero as well as its index in the array
[cos_min,cos_index] = min(cos_alpha_raw_mean_avg_abs);
%look at the real value of the element with that index
cos_zero_approx = cos_alpha_raw_mean_avg(cos_index,1);
%raw_line_voltage_result = Voltage(cos_alpha_raw_line_avg_index,1);

%if it is above zero read the next element(below zero) and interpolate
if (cos_zero_approx>=0)
    %check if the next element exists
    if (cos_index+1>sets)%if not interpolate with the previous element
        left_cos = cos_alpha_raw_mean_avg(cos_index-1,1);
        left_voltage = Voltage(cos_index-1,1);

        right_cos = cos_alpha_raw_mean_avg(cos_index,1);
        right_voltage = Voltage(cos_index,1);

    else%if yes interpolate with the next element
        left_cos = cos_alpha_raw_mean_avg(cos_index,1);
        left_voltage = Voltage(cos_index,1);

        right_cos = cos_alpha_raw_mean_avg(cos_index+1,1);
        right_voltage = Voltage(cos_index+1,1);
    end
end

else
    %check if the previous element exists
    if (cos_index-1<0)%if not interpolate with the next element
        left_cos = cos_alpha_raw_mean_avg(cos_index,1);
        left_voltage = Voltage(cos_index,1);

        right_cos = cos_alpha_raw_mean_avg(cos_index+1,1);
        right_voltage = Voltage(cos_index+1,1);

    else%if yes interpolate with the previous element
        left_cos = cos_alpha_raw_mean_avg(cos_index-1,1);
        left_voltage = Voltage(cos_index-1,1);

        right_cos = cos_alpha_raw_mean_avg(cos_index,1);
        right_voltage = Voltage(cos_index,1);
    end
end

end
raw_mean_voltage_result = left_voltage + (right_voltage-left_voltage)*(0-
left_cos)/(right_cos-left_cos);

%% Calculate parameters for next iteration

%read the result from the previous iteration
V_prev=raw_mean_voltage_result;
min_step_prev = Voltage_step;

```

```

%calculate new step size
min_step_next = min_step_prev/step_change_factor;

if (min_step_next<= min_step_absolute)

%    return;
end

%calculate the parameters for the new iteration

%calculate the V_min parameter
V_min_next = V_prev*lower_limit;
%rounds down to the nearest min_step multiple
V_min_next = int32(V_min_next/min_step_next);
V_min_next = cast(V_min_next, 'double')*min_step_next;

if (V_min_next<=Down_range)
    V_min_next=Down_range;
end

%calculate the V_min parameter
V_max_next = V_prev*upper_limit;
%rounds up to the nearest min_step multiple
V_max_next = int32(V_max_next/min_step_next);
V_max_next = cast(V_max_next, 'double')*min_step_next + min_step_next;

if (V_max_next>=Down_range)
    V_max_next=Up_range;
end

%% Print

%*****Print*****
desired_set = 5;

% *****Col data*****
I_1_line(:,1)=I_data(:,desired_col,1,desired_set);
I_2_line(:,1)=I_data(:,desired_col,2,desired_set);
I_3_line(:,1)=I_data(:,desired_col,3,desired_set);
I_4_line(:,1)=I_data(:,desired_col,4,desired_set);
I_5_line(:,1)=I_data(:,desired_col,5,desired_set);

% *****Full raw data*****
I_1_raw=I_data(:, :, 1, desired_set);
I_2_raw=I_data(:, :, 2, desired_set);
I_3_raw=I_data(:, :, 3, desired_set);
I_4_raw=I_data(:, :, 4, desired_set);
I_5_raw=I_data(:, :, 5, desired_set);

cos_alpha_raw_line_plot = cos_alpha_raw_line(:,desired_set);

```

```

cos_alpha_raw_mean_plot = cos_alpha_raw_mean(:,desired_set);

cos_alpha_data_raw_plot = cos_alpha_data_raw(:, :, desired_set);

Rows = 1:1:Col_range;

%****Print final result ****

figure(5)
subplot(2,1,1)
plot(Voltage, cos_alpha_raw_line_avg, '-+g');
hold on
plot(raw_line_voltage_result,0, '+r');
title('cos alpha raw line avg');
grid on;
%axis equal;
%axis tight;

subplot(2,1,2)
plot(Voltage, cos_alpha_raw_mean_avg, '-+g');%print cos voltage
hold on
plot(raw_mean_voltage_result,0, '+r');%print cross point
hold on
plot(V_min_next,0, 'ob');%print new min voltage
hold on
plot(V_max_next,0, 'ob');%print new max voltage
title('cos alpha raw mean avg');
grid on;
%axis equal;
%axis tight;

```

Appendix D: Micro Mirrors Experimental Setup

The setup used for the micro mirror experiment uses a computer, the driving computer, along with power supply and driver hardware to control the position of the micro-mirrors. A second computer, the measuring computer, is connected to the interferometric camera and used to detect the displacements. A software interface provided by Boston Micromachines was used on the driving computer. Full specifications for all hardware used are shown in Table 7. Detailed instructions for the operation of the micro mirror driver are outlined in an instruction manual which accompanies the hardware.

Table 7: Micro mirror experiment hardware specifications

Component Name	Manufacturer	Part Number	Serial Number
Power Supply	GW INSTEK	GPR-30H100	EI816833
Driver	Boston Micromachines	Mini-DM-HS	080418DB76
Micro mirror array	Boston Micromachines	μ SLM036-300-AI	11W284#038-300S16
Serial expansion card	National Instruments	PCI-6713	10416D21

A GWINSTEK power supply shown in Fig. 43(a) was used to power the micro-mirror driver, Fig. 43(b). This driver was connected to the National Instruments serial expansion card in the driving computer through a serial cable, and also to the micro mirror array through a parallel IDE cable.

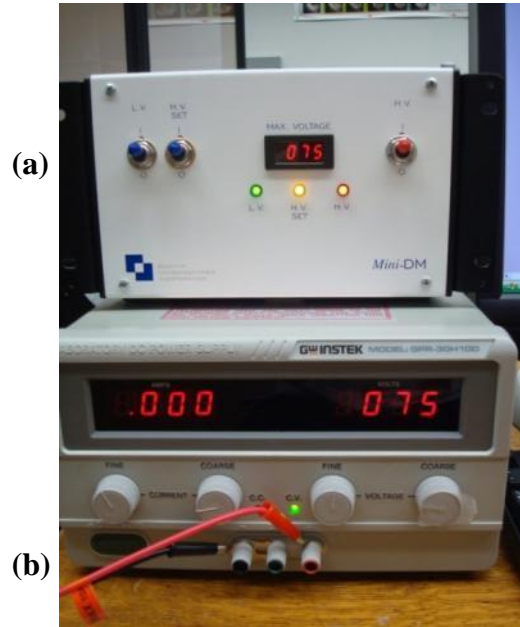


Fig. 43. The micro mirror driving hardware.

The micro mirror array was fastened to the five axis positioning station, shown in Fig. 46, under the interferometer to allow for proper orientation and alignment.

Fig. 44 shows the micro mirror array connected to the IDE connector board. Fig. 45 shows only this board. Note the clamp, which is upright. Once the chip is inserted properly into the connector board, this clamp must be pressed down to make sure the pins are engaged.

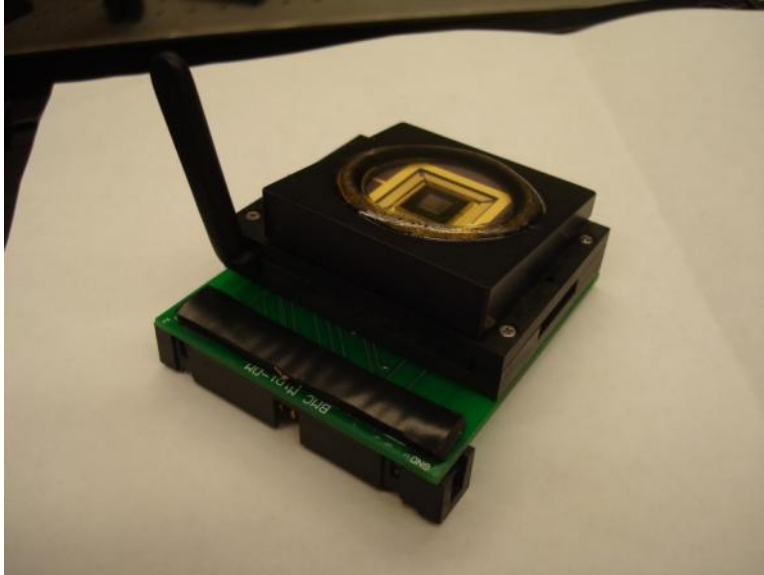


Fig. 44. Micro mirror array inserted into IDE connector.

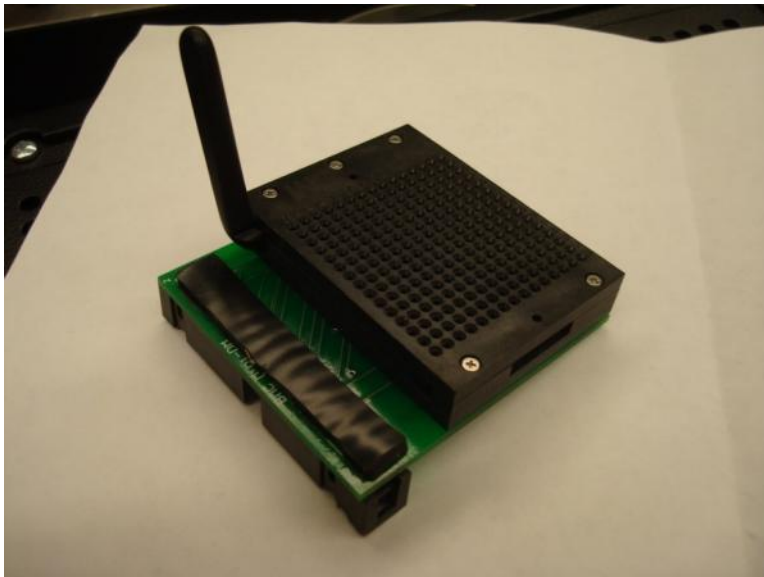


Fig. 45. IDE Connector without micro mirror array.

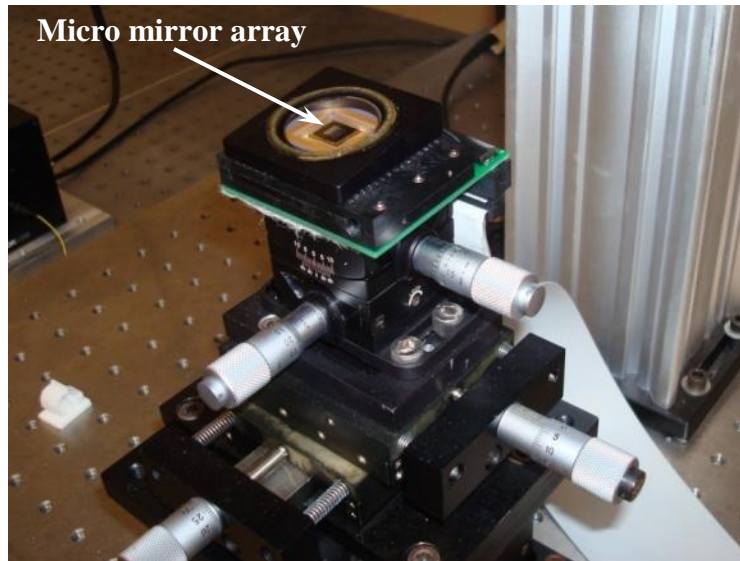


Fig. 46. Positioning the micro mirror array.

The software interface provided controls the center six by six mirrors. The basic functions of the software allow for the direct application of voltage to individual mirrors in the array. Multiple mirrors can be controlled at once, and the position of mirrors can be controlled incrementally. Fig. 47 shows six mirrors being driven at 75 volts to create the letter "J." An image captured of this configuration using the interferometric camera is shown in Fig. 10 on page - 18 -.

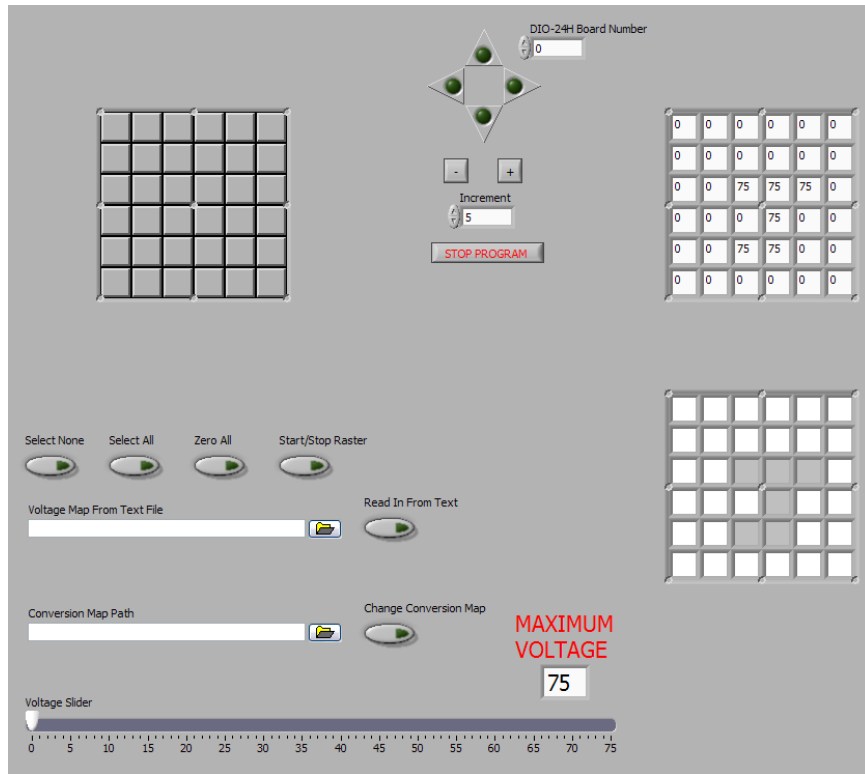


Fig. 47. Micro mirror interface software.

This software was also capable of exporting configurations such as the "J" for later use. Additional features are included in the software that were not used for our testing.

Appendix E: Commercial Components Used in Thermal Loading Station

This appendix specifies the commercially available components that were used in the fabrication of the thermal loading station.

Thermistor Omega 44005

The Steinhart-Hart Equation has become the generally accepted method for specifying the resistance vs. temperature characteristics for thermistors. The Steinhart-Hart equation for temperature as a function of resistance is as follows:

$$\frac{1}{T} = A + B[\ln(R)] + C[\ln(R)]^3, \quad (E1)$$

Where, A, B and C are constants derived from 3 temperature test points.

R = Thermistor's resistance in Ω

T = Temperature in degrees K

The A, B and C constants for this model are as follows:

$$A = 1.403 \times 10^{-3}$$

$$B = 2.373 \times 10^{-4}$$

$$C = 9.827 \times 10^{-8}$$

Fig. 48 is a plot of the resistance of the thermistors with respect to temperature in the temperature range that they were used as well as prediction of the resistance based on the conversion formula.

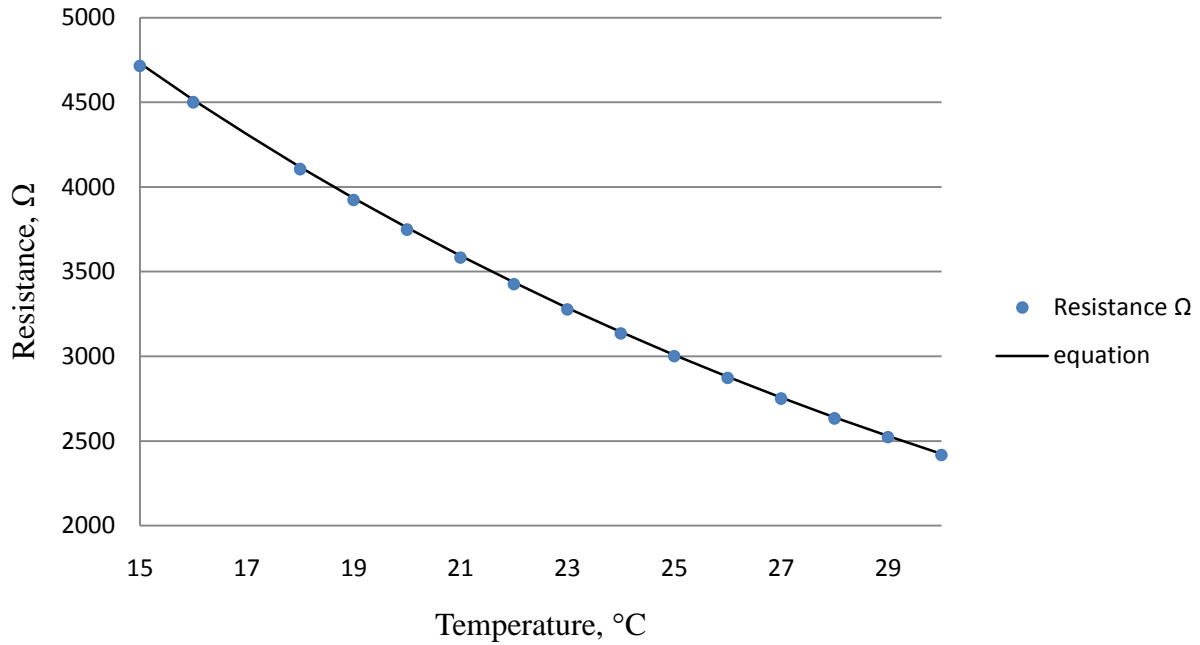


Fig. 48. Thermistor sensitivity.

The resistance of one of the thermistors was measured directly with a multi-meter in 2-wire resistance measurement mode. Fig. 49 shows a wiring diagram of the temperature measurement setup:

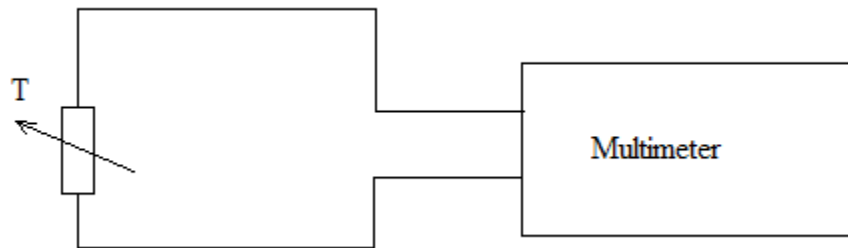


Fig. 49. Thermistor wiring diagram.

Thermo-electric couple CUI, inc. model CP85138

Fig. 50 shows the thermal characteristic of the TEC used in the setup:

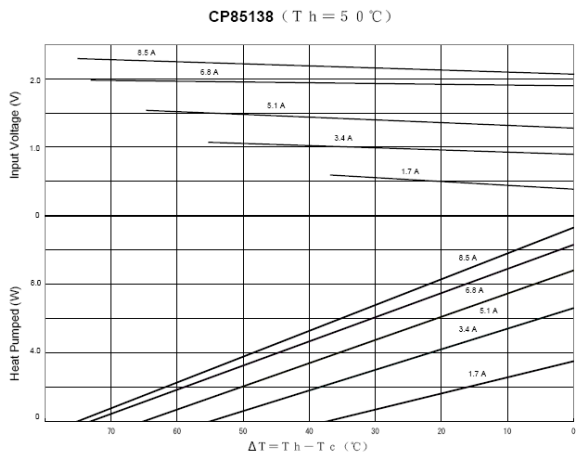
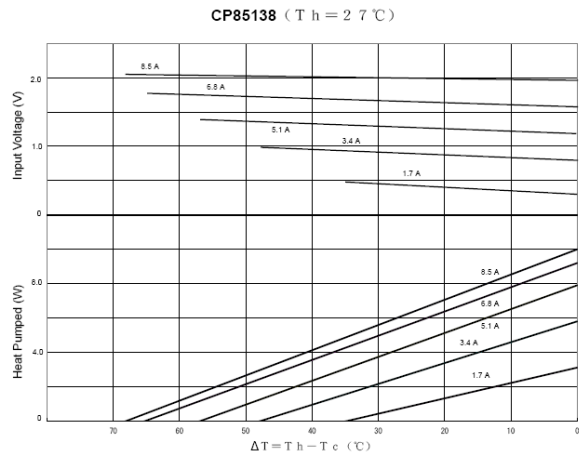


Fig. 50. TEC performance graph.

Fig. 51 is a wiring diagram of the TECs in the setup:

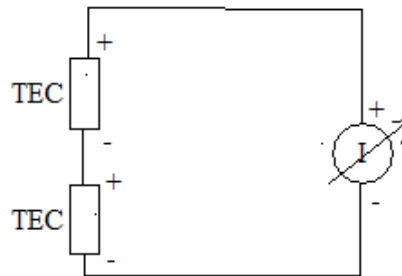
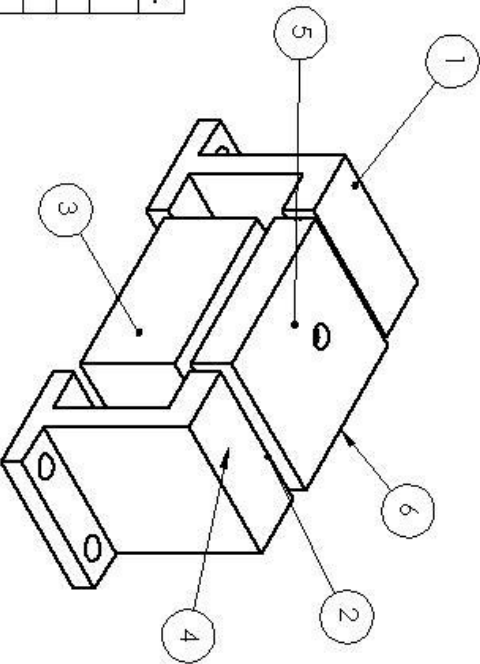


Fig. 51. TEC wiring diagram.

Appendix F: Mechanical drawings of fabricated components

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	ds1-02	Rapid Prototype Part	2
2	ds1-01	Thermal Plate	1
3	ds1-03	Heat Sync	1
4	97334A199	Master Carr	4
5	Array	Sample Part	1
6	ds1-04	Cover	1



TOLERANCES:
 FRACTIONAL 1:05
 DECIMAL:
 ANGULAR:
 HOLE 1:1
 BEND 1:1
 HOLE 1:1
 TYPICAL 1:02
 HOLE 1:1
 DECIMAL 1:005

DATE: 02/25/09

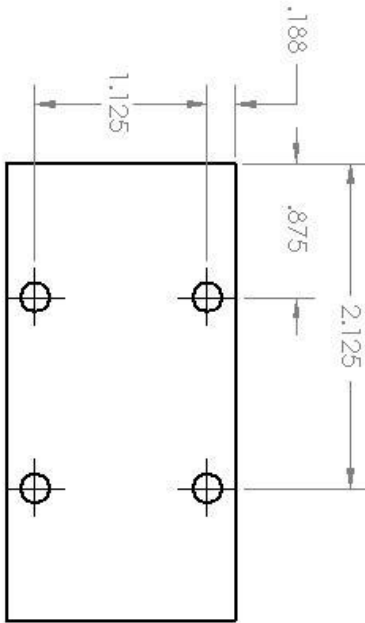
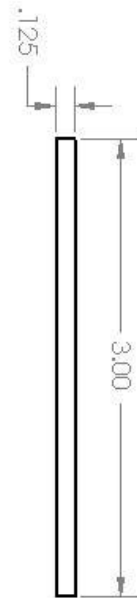
WPI
 DEPARTMENT OF MECHANICAL ENGINEERING
 WORCESTER POLYTECHNIC INSTITUTE

FULL ASSEMBLY

TITLE: **FULL ASSEMBLY**

MATERIAL: Aluminum
 PROJECTION: Third Angle

SIZE: A
 SCALE: 1:4
 CAGE CODE: 81359
 WEIGHT:
 DRAWN BY: Ryan Fossett
 SHEET 1 of 4



TOLERANCES:
 FRACTIONAL 1.05
 ANGULAR:
 MACH ±.1
 HORN ±.1
 TPO PLAGE
 DECIMAL 1.02
 THREE PLACE
 DECIMAL 1.005

All dimensions
 are inches
 and degrees

PROJECTION:
 Third Angle

MATERIAL:
 Aluminum

Date:02/25/09



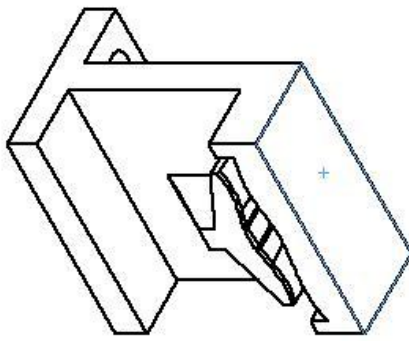
DEPARTMENT OF
 MECHANICAL ENGINEERING
 WORCESTER POLYTECHNIC INSTITUTE

TITLE


THERMAL PLATE

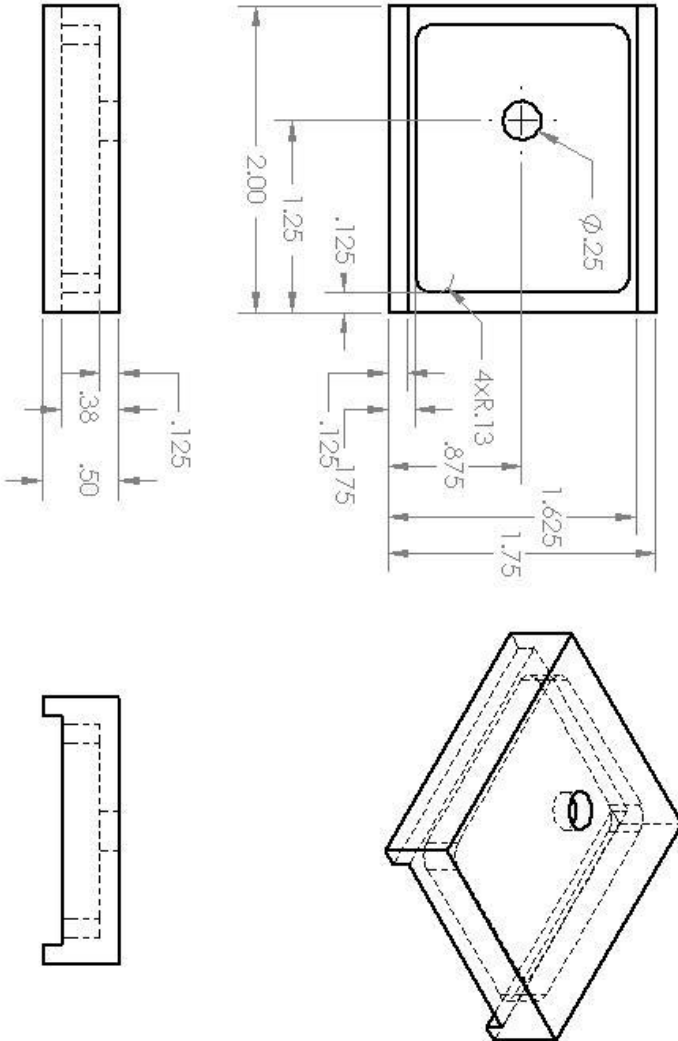
SIZE A CAGE CODE 81359 DRAWN BY: Ryan Fossett

SCALE 1:1 WEIGHT SHEET 2 OF 4



Rapid Prototype

<p>TOLERANCES: FRACTIONAL ±.05 ANGULAR: MACH ±.1 BEND ±.1 TWO PLACE DECIMAL ±.02 THREE PLACE DECIMAL ±.005</p>		<p>All dimensions are inches and degrees</p>		 <p>DEPARTMENT OF MECHANICAL ENGINEERING WORCESTER POLYTECHNIC INSTITUTE</p>	
<p>PROJECTION: Third Angle</p>		<p>TITLE</p> <p>RAPID PROTOTYPE</p>		<p>DATE: 02/25/09</p>	
<p>MATERIAL:</p>		<p>SIZE A</p>		<p>SCALE 1:1</p>	
<p>DRAWN BY: Ryan Fossett</p>		<p>CAGE CODE 81359</p>		<p>WEIGHT</p>	
<p>SHEET 3 OF 4</p>		<p>WPI</p>		<p>81359</p>	



TOLERANCES:
 FRACTIONAL 1-.06
 DECIMAL:
 .01
 .02
 .03
 .04
 .05
 .06
 .07
 .08
 .09
 .10
 .12
 .15
 .20
 .25
 .30
 .38
 .50
 .63
 .80
 1.00
 1.25
 1.50
 2.00
 2.50
 3.00
 4.00
 5.00
 6.00
 8.00
 10.00

All dimensions are inches and degrees
 PROJECTION: Third Angle
 MATERIAL: Plexi-glass



DEPARTMENT OF MECHANICAL ENGINEERING
 WORCESTER POLYTECHNIC INSTITUTE

COVER

Date: 02/25/09

SIZE: A
 SCALE: 1:1

Cage Code: 81359
 WEIGHT:

DRAWN BY: Ryan Fossett
 SHEET 4 OF 4