



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Real-time measurement of biaxial tensions using digital image correlation methods

Haemish Kyd

Thesis presented for degree of:

Msc (Eng) in the Department of Mechanical Engineering

Faculty of Engineering and Built Environment

University of Cape Town (UCT)

2022

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

Declaration

I hereby grant the University of Cape Town free license to reproduce for the purpose of research, either the whole or any portion of the contents in any manner whatsoever of the above dissertation.

I know the meaning of plagiarism and declare that all the work in this document, save for that which is properly acknowledged, is my own. I also declare that this material has not been submitted for any purpose or examination to any other Department or University.

Signed:

Date:

Signed by candidate

13 February 2022

Haemish Kyd
Mr

Abstract

The mechanical properties of biological materials need to be measured for various applications. A means of inducing biaxial tensions in samples like these is with an inflation or bulge test. Normally the material under test would be measured with displacement gauges, however, under these conditions, where the specimen is soft and further, where the measurement cycle cannot be reliably paused, a contactless real-time measurement system is necessary to obtain reliable deformation data.

Digital Image Correlation (DIC) is one such method. Pioneered in the 1980s the field has developed from basic 2D displacement measurements to very sophisticated full field 3D displacement measurement systems.

The question becomes can the current state of the field, as well as the advances in modern technology, be leveraged to create a useable 3D DIC measurement system that is:

- Useable in a real-time context.
- Portable enough to be able to run these experiments wherever the experiment apparatus is located.
- Cost effective enough to reduce the barrier to entry that the current commercial options present.

To this end off-the-shelf components were acquired to form the technology base of the system. The open-source DICE framework, which enabled the necessary level of access to the underlying code base, was implemented on an NVIDIA Jetson Nano single board computer. Synchronised, stereo image acquisition was implemented via an Arducam 12 MP camera system. A stepper motor controlled linear drive was used to experimentally investigate accuracy and speed of the DIC system, for both rigid body motion and deforming targets.

A thorough review of the concepts involved in DIC is undertaken followed by a detailed description of the design and build of the system.

Ultimately a set of experiments are executed that show that, within a set of important constraints, it is indeed possible to run 3D DIC in real-time with off the shelf, cost effective components.

Acknowledgements

Undertaking something of this nature, at a later time in one's life, requires a lot of support and understanding from those around you. These people need to be acknowledged.

When the idea of this project first presented itself, I was sceptical that I had the time or ability to do it justice. It was through Dr Govender's support and encouragement that I felt I could do this, and it has been through his facilitation and support that I have. He will always underplay his role but his guidance and understanding in me navigating my full-time job in conjunction with an MSc, has been essential in getting me through this project.

I would like to also recognise my co-supervisor, Prof. Nicolls, whose comments, and interventions have been appreciated. His third person perspective helped when I had become too close to the work.

To all the staff of the various engineering departments and workshops at the University of Cape Town that I have briefly interacted with, I have appreciated all of the support.

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged and appreciated. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

As mentioned, I have a full-time job and my company has supported me with time and financial assistance throughout this project. Without their support this would not have been possible.

To my friends in academia, Debbie and Reuben, who have been down this path before, your words of advice, points of correction and commiseration during the less pleasant moments were more appreciated than I probably expressed. Thank you for always checking in on my progress and probably hyperbolising how well I was doing.

To my parents, siblings and in-laws of all flavours, you are the embodiment of the newly coined adage – it takes a village to raise a master's student. Thank you for all of your encouragement.

Lastly to my wife Mary. You agreeing to me doing this project, with not a bit of hesitation, had a big part to play in my decision to actually do it. The weekends where I have spent most of the time in the study while you've run around with the kids is just a sample of the compromises you have made to facilitate this. You'll get your husband back soon, whether you want him or not. Thank you for always listening to my technical rambling and being ever impressed even if what I was showing you was just a screen of numbers.

Cameron and Isabella, you won't remember your dad doing this, but I hope that it will drive you to realise that all the excuses you use to not do things, are just that.

Table of Contents

Declaration.....	2
Abstract	3
Acknowledgements	4
1. Introduction.....	10
2. Literature Review	14
2.1. Introduction	14
2.2. Theoretical Basis.....	14
2.2.1. Basic Principle.....	14
2.2.2. Cross Correlation Criteria	20
2.2.3. Optimisation Algorithm	23
2.3. Chronology of Developments.....	27
2.4. 3D DIC Theory	29
2.4.1. Camera Calibration	29
2.4.2. 3D Calibration Specifics	31
2.4.3. 3D DIC Process	32
2.5. Efficiency Improvements	34
2.5.1. Improvements in Methods, Preparation and Tools.....	34
2.5.2. Algorithmic Improvements	36
2.5.3. Real Time DIC.....	38
3. Design Considerations.....	39
3.1. Hardware Design	39
3.2. Software Design.....	42
3.2.1. Open-Source Software Choices.....	42
3.3. Integration and Build	45
3.3.1. Code Development.....	45
3.3.2. Experimental Platform	46
3.4. Design Description.....	52
3.4.1. DICe Interface Logic	54
3.4.2. Image Capture.....	55
3.4.3. User Interface	57
3.4.4. Calibration Logic	58

4.	Experimental Data.....	61
4.1.	Experiment 1 – Noise Floor Experiment.....	64
4.1.1.	Description	64
4.1.2.	Procedure.....	64
4.1.3.	Results	65
4.2.	Experiment 2 – Speed: Number of Subsets.....	68
4.2.1.	Description	68
4.2.2.	Procedure.....	68
4.2.3.	Results	69
4.3.	Experiment 3 – Speed: Subset Size	71
4.3.1.	Description	71
4.3.2.	Procedure.....	71
4.3.3.	Results	72
4.4.	Experiment 4 – Motion (Accuracy).....	76
4.4.1.	Description	76
4.4.2.	Jig Motion Confirmation.....	76
4.4.3.	Two-Dimensional Displacement	78
4.4.4.	Three-Dimensional Displacement - Stepped Profile	81
4.4.5.	Three-Dimensional Displacement - Slow Continuous Profile	85
4.5.	Experiment 5 – Motion (Speed)	87
4.5.1.	Description	87
4.5.2.	Procedure.....	87
4.5.3.	Results	87
4.6.	Experiment 6 – Bulge Test.....	90
4.6.1.	Description	90
4.6.2.	Procedure.....	90
4.6.1.	Results for larger indenter	93
4.6.2.	Results for smaller indenter.....	98
5.	Concluding Discussion	103
5.1.	Feasibility Discussion	103
5.1.1.	Summary	103
5.1.2.	Results Summary	103
5.1.3.	Comments on Feasibility.....	104

5.2.	Potential Further Work.....	105
5.3.	Concluding Remarks.....	105
6.	References.....	106
7.	Appendices.....	113
7.1.	Appendix 1	113
7.2.	Appendix 2	114
7.3.	Appendix 3	116
7.4.	Appendix 4	116
7.5.	Appendix 5	117

Table of Figures

Figure 1: Typical Bulge Test Setup.	10
Figure 2: PDP-8/E.....	11
Figure 3: 3D DIC Setup.....	12
Figure 4: Logical Flow of DIC Process.	15
Figure 5: Displacement Measurement Example.....	16
Figure 6: Undeformed Image.....	17
Figure 7: Deformed Image.	17
Figure 8: Pixel Intensity Levels.....	17
Figure 9: Correlation Values.	17
Figure 10: Example Speckle Pattern.....	18
Figure 11: Pixelated and Linear Interpolated.	18
Figure 12: Cubic Interpolation Between Pixels.....	19
Figure 13: Cross Correlation Process on an Image.....	20
Figure 14: Output of Cross Correlation Process.....	20
Figure 15: Region of Interest and Subset Allocation.....	21
Figure 16: Normalised Cross Correlation Results.	23
Figure 17: Subset deformation.....	24
Figure 18: Bilinear Interpolation Pixel Map.....	25
Figure 19: 3D-DIC Experimental Setup.....	29
Figure 20: Epipolar Geometry Concept Diagram.....	31
Figure 21: Schematic showing the principles of stereo-DIC [40].	32
Figure 22: Feature matching between two frames.....	33
Figure 23: Single Camera Stereo DIC Experimental Setup.....	36
Figure 24: Magnitude Spectrum of Speckle Pattern.	37
Figure 25: DICE Software Architecture.	44
Figure 26: Camera Orientation for Different Axes Measurements.	46
Figure 27: Experimental Design Setup.	47
Figure 28: Test Jig with Indenter.....	48
Figure 29: Final Experimental Hardware Setup.	49
Figure 30: Test Jig Program Flow.	50
Figure 31: Final Software Architecture.....	52
Figure 32: Final Code Flow.....	53
Figure 33: Arducam 12MP Synchronized Stereo Camera.	55
Figure 34: Gstreamer pipeline development [73].....	56
Figure 35: Final User Interface.....	57
Figure 36: Calibration capture process (showing analysis of corners).	60
Figure 37: Frame of Reference for Experiment.....	61
Figure 38: Experimental Roadmap.	62
Figure 39: Illustration of subsets chosen for synchronised cameras (run 2).	65
Figure 40: Graphs of motion of each subset in each axis (Run 1).	65
Figure 41: Graphs of motion of each subset in each axis (Run 2).	66

Figure 42: Processing Time for Varying Numbers of Subsets.	69
Figure 43: Region of Interest Selector - the blue dots were used.	72
Figure 44: Processing Time for Varying Subset Sizes.	73
Figure 45: Z Displacement Response with Varying Subset Sizes.	74
Figure 46: Heat map for subset sizes 15 and 7 pixels.	74
Figure 47: Pixel Density Comparison	75
Figure 48: Y Displacement Measurement Equipment Setup.	78
Figure 49: Y displacement measured by 2D DIC.	79
Figure 50: Axis Alignment Analysis.	80
Figure 51: Displacement Measurement Equipment Setup	81
Figure 52: X/Y Positive Frame of Reference	81
Figure 53: X/Y Negative Frame of Reference.	81
Figure 54: Z Displacement with Stepped Motion.	82
Figure 55: Positive and Negative Frame of Reference.	84
Figure 56: Z Displacement with continuous motion.	85
Figure 57: Speed set at 0.8mm/s and 1.4mm/s.	88
Figure 58: Speed set at 5mm/s.	89
Figure 59: 25mm indenter installed into system.	91
Figure 60: Latex bulge due to indenter.	91
Figure 61: Illustration of the Estimation of the Curve.	92
Figure 62: Example of the filtering algorithm.	94
Figure 63: Progression of Bulge Test.	95
Figure 64: Extract of the data representing the bulge – large indenter.	96
Figure 65: Bulge estimation – large indenter.	96
Figure 66: Progression of Bulge Test.	99
Figure 67: Extract of the data representing the bulge – small indenter.	100
Figure 68: Bulge estimation – small indenter.	100
Figure 69: Smaller Subset Size Tracking	101
Figure 70: Estimate of radius with smaller subsets used.	102

1. Introduction

The mechanical properties of biological materials need to be measured for various applications. For biological membrane tissues, a popular method of testing is the inflation or bulge test [1] as shown in Figure 1. Given the nature of these materials (i.e., they are generally soft and fragile) and that testing in interrupted steps (i.e., deform, pause, measure, continue) is undesirable, a contactless real-time measurement system would be ideal to obtain this deformation data. Real-time data can also be extremely beneficial since the tests can be interrupted prior to significant specimen damage, facilitating post-mortem microscopy studies. This project attempts to understand the implications, constraints, and difficulties in creating such a system.

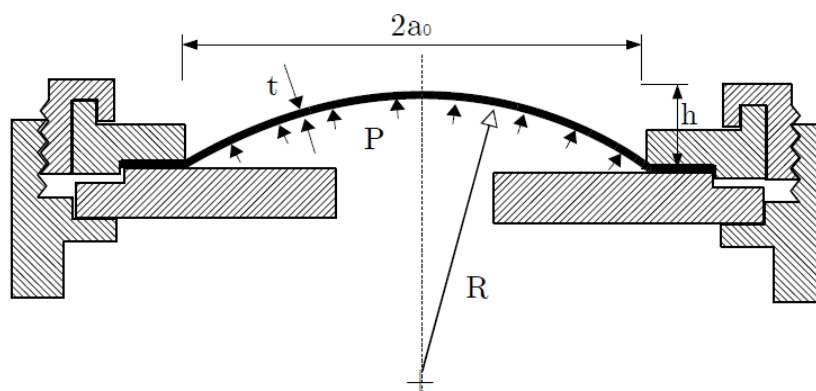


Figure 1: Typical Bulge Test Setup.

Historically many methods were accepted as standard for the optical contactless measurement of displacement and strain in experimental stress analysis. These include holography, speckle interferometry, speckle photography and white light speckle [2]. Many of these optical measurement techniques suffer from issues of stability and complexity of calculation. In the early 1980s, with the wider acceptance of computer technology, the concept of Digital Image Correlation (DIC) started being applied to the problem of contactless displacement and strain measurements [3].

Sutton et al. [3] defined the basic premise of DIC in the one of the first papers to cover the field. In this paper they claim:

“It has been stated many times that the computer will alter the way we live. Without question, this simple statement continues to be proven true every day. However, the full potential of the computer has not been realized in many areas. In particular, those people who wish to contribute to a better understanding of our world through innovative measurements have seen little change.”

The basic premise of DIC was defined in the paper above. Even though this statement was made in the early 1980s, when computers were just coming into common use, it remains oddly applicable today.

Following Sutton’s early contribution, many improvements were developed, specifically with regards to accuracy and speed. These have mainly been attributed to improved mathematical approaches [4],[5],[6], optimised correlation criteria [7] and better experimental constraints [8],[9]. The discussion around advances in the computer hardware used for these experiments, the huge change seen in the software landscape, and the influence it can have on the way DIC is done, has not been explored in any depth.

The first experiments were done in the early 1980s using a PDP-8/E, shown in Figure 2, and VAX minicomputers. It is worth noting this “minicomputer” was a full cabinet and sometimes more depending on the degree of data storage. The PDP-8 series of computers had a processing speed of about 1MHz contrasted against a basic contemporary smartphone currently running at a speed of 2Ghz.



Figure 2: PDP-8/E.

Specifically, due to the requirements of the self-driving car industry, standalone, incredibly powerful computers now come in very portable packages [10]. This industry has demanded the availability of computers that are compact (since they must fit into the current concept of a car) but powerful enough to run image recognition and artificial intelligence algorithms.

The recent surge in the open-source community also means that high quality software is now available with GPL, MIT or other permissive licenses [11]. In the last 10 years the open-source community has seen an uptick in the number of high-quality products available. These include RabbitMQ (a message queueing system), Python, Golang and Kubernetes to name a few. The space of image processing has not been neglected with OpenCV becoming the ubiquitous image processing code base. Digital image correlation has also benefitted, and a few high-quality code bases are now available. These options will be elaborated on in later chapters.

With all this in mind, the questions then becomes whether these improvements in software algorithms and hardware processing power have made real-time, 3D DIC deformation measurements possible and whether this can be performed in a cost-effective way?

The process of DIC has changed little from those initial papers introduced in the early 1980s. A specimen is deformed with the desired loads and boundary conditions, while cameras record the process. This sequence of images is then fed into a computer. The areas of interest are defined and the computer analyses the data and produces a set of data that represents the displacement and strain of the system. The cameras are usually using specialised lighting. The

computer is usually a robust machine, which including commercial DIC software, comes in at a non-trivial cost [12].

This project originated from an application where 3D deformation is unavoidable and not negligible, rather than the simpler case of planar, 2D deformation. Running these algorithms in a 3D context involves running the 2D algorithms on two cameras looking at the same target (as shown in Figure 3), after first determining the orientation of the two cameras to each other in space, using a calibration target. The difference in perspective of the two cameras allows for a 3D representation of the data [13]. This poses more challenges computationally.

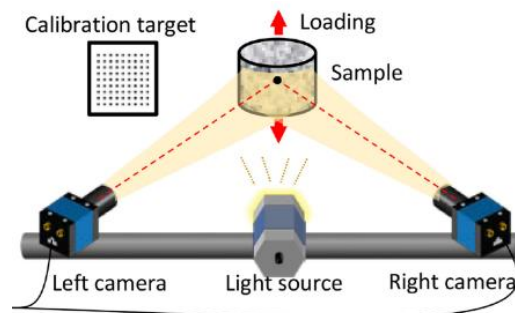


Figure 3: 3D DIC Setup [14].

There is very little doubt that the desire for near real-time processing will yield results which are not as accurate as the usual DIC algorithms that are post-processed with little consideration for speed. The primary research questions are:

- Given a specified image resolution and frame rate, can a 3D DIC system yield acceptable accuracy of deformation measurements when running in near real-time?
- What implications arise from the constraint that widely available and cost-effective hardware and software be used?

The objectives of the project are:

- Develop software to run 3D DIC using open-source software, tools and libraries.
- Run this software using commercially available, off-the-shelf cameras and components.
- Port this software so that it is running on a commercially available, portable, single-board computer.
- Create an experimental setup to test the efficacy of this system with regards to speed and accuracy.¹

The first part of this project will review the current state of the field highlighting the efficiencies introduced in the mathematics and processes. The process of the development of the code base, the open-source code used and the porting to the single-board computer will then follow.

¹ Initially testing would have been carried out in the UCT laboratories, using tensile test machines of known accuracy but due to the unique restrictions of the COVID19 pandemic these labs were inaccessible. A standalone test rig was therefore created for testing.

Finally, a description of the experimental setup and a study of the results achieved using this platform will be presented.

2. Literature Review

2.1. Introduction

DIC is not an overly complicated process from a mathematical perspective, but to someone unfamiliar with the fundamentals, the concepts can seem counterintuitive and inaccessible. In the following few sections, the principles involved in DIC have been described with the intention of removing the veneer of complexity.

The theoretical basis for the 2D DIC algorithm is covered first. This initial discussion will cover the basic principle and then attempt to extend this first principles discussion to cover the actual theoretical implementation of the concept and its optimisation. Following this a chronological presentation of major achievements in the field of DIC is covered. Once the concept of 2D DIC is understood this can be extended to two cameras and 3D DIC can then be covered in some detail. Since 3D DIC uses two cameras, a discussion on camera calibration theory is covered before an elaboration on the actual 3D DIC process.

In the development of DIC many improvements have been made to the technology. These are too numerous to exhaustively cover but since they will be key to understanding the limits of real-time DIC measurement, an attempt is made to cover those efficiency improvements that are relevant to this project. The efficiency improvements covered will be research into the environment of the experiment (e.g., lighting and target preparation) as well as algorithmic improvements. The chapter will finally cover the current state of real-time DIC.

2.2. Theoretical Basis

2.2.1. Basic Principle

DIC works by comparing images of a specimen at various stages of deformation. Within the reference (un-deformed) image a region of interest (ROI) is chosen, and that ROI is then split into equally sized subsets. These subsets are then tracked through stages of the deformation and from this, 2D or 3D deformation vector fields and strain maps can be measured [15].

The basic steps are highlighted in Figure 4 [16].

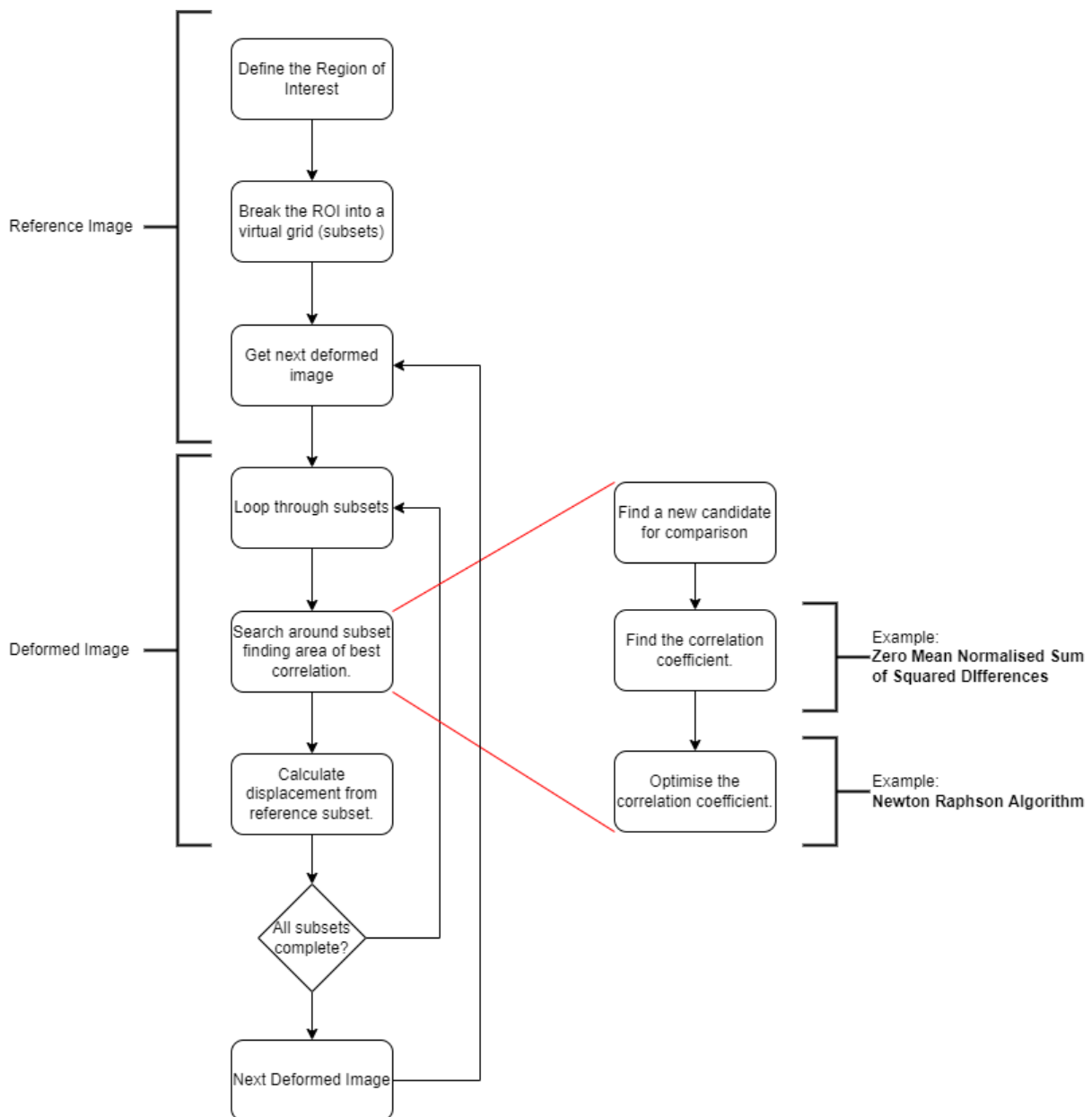


Figure 4: Logical Flow of DIC Process.

To understand this process at its most fundamental level, let us assume that we have a surface with a prominent feature painted onto it. Now let us assume that this surface was deformed or (as in Figure 5 below) translated in the plane of the surface. DIC attempts, with a camera and an algorithm running on a computer (usually), to tell us how much this prominent feature has moved.

In Figure 5, the camera is stationary, and the positions change such that X and X' are two frames taken before and after the movement of the feature.

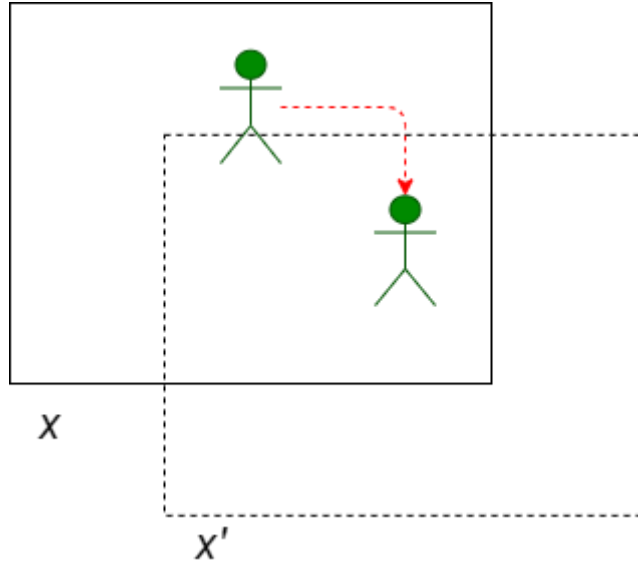


Figure 5: Displacement Measurement Example.

From X , the aim would be to identify the intensity level of the pixels around the feature and find the same set of intensity levels of pixels in X' . Once these are identified in X' , the displacement in both directions can be determined. The area of interest around the feature is referred to as a subset [12]. It is worth noting that this is a very fundamental example. For DIC, as explained in section 2.2.2, the target is speckled and broken into adjacent subsets. These subsets are *all* individually tracked in this way and as such we can get a full-field measurement of the displacement.

To determine the matching intensity levels in X and X' , a correlation coefficient is used. An example is given below:

$$C(x, y, x^*, y^*) = \frac{\sum F(x, y)G(x^*, y^*)}{\sqrt{\sum F(x, y)^2 \sum G(x^*, y^*)^2}}$$

Equation 1: Generic Correlation Function.

where $F(x, y)$ denotes the pixel intensity level in X and $G(x^*, y^*)$ denotes the pixel intensity level in X' . The values are summed across the subset and a maximum value (or minimum value depending on the criteria used) for C is obtained when the intensities inside the subsets are closest. The displacements are determined when the correlation coefficient is at an extremum.

Equation 1 represents an option for the cross-correlation criteria. There are a few options that have been considered in the literature and these are discussed in more detail in Section 2.2.2.

What follows is a trivial example that covers the concept of matching a subset via cross correlation.

If we assume a greyscale image of a single row of pixels:



Figure 6: Undeformed Image.

We now translate this image by 3 pixels:



Figure 7: Deformed Image.

The graphical representation of these grey scale values is as follows:

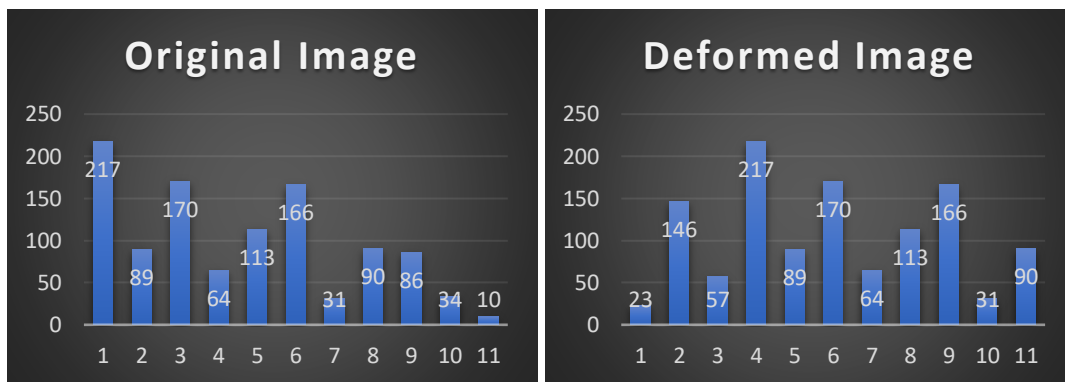


Figure 8: Pixel Intensity Levels.

It is trivial, looking at the values, to see that the entire pattern has shifted by three pixels to the right. Let us choose a subset which will include the first six pixels in the original image. Using Equation 1 we will determine the correlation co-efficient in a moving window across the deformed image. The results are represented by the following graph:

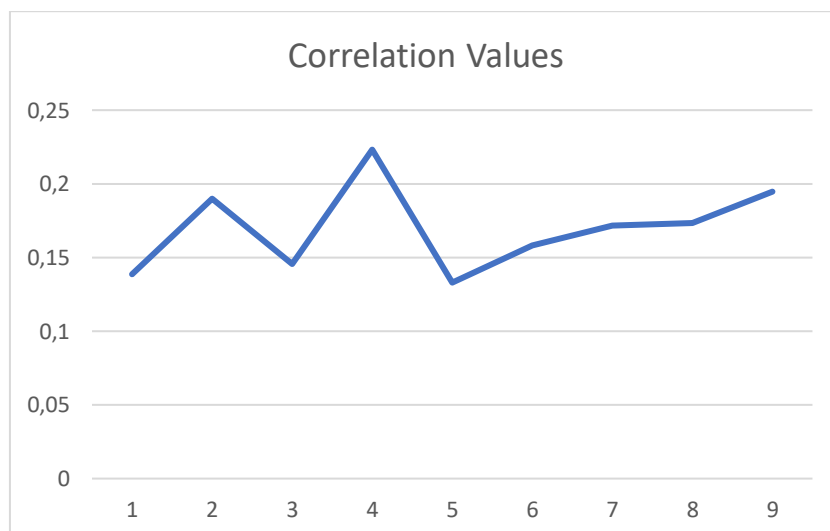


Figure 9: Correlation Values.

The graph peaks at position four – which means that the sample window (subset) correlates most at this point. The image has therefore been displaced by 3 pixels. The detailed calculations

can be seen in Appendix 1. While this example is somewhat contrived and simplistic, it highlights the process in a way that is (hopefully) understandable.

In DIC the target tracked is usually some sort of a speckle pattern. This speckle pattern has to create a target that is both extractable (i.e., it must be able to be seen by the camera) but also be random enough to not create correlation confusion. An example of a speckle pattern is shown in Figure 10.

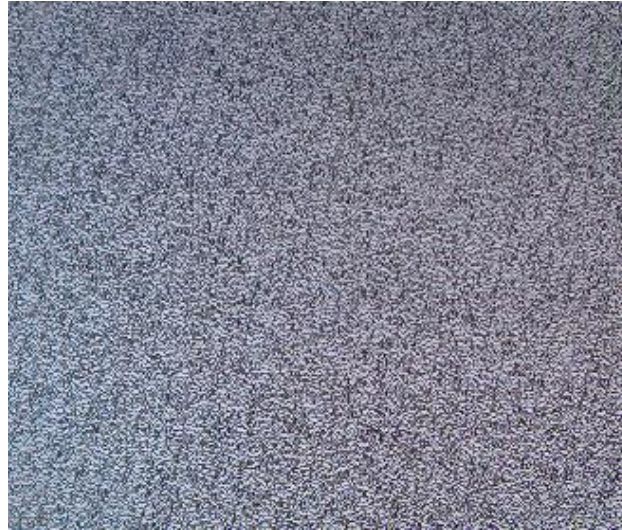


Figure 10: Example Speckle Pattern.

The method described thus far relies on comparing data at discrete pixel points. This level of accuracy is not sufficient since displacements do not coincide with integer pixels [12]. To overcome this limitation, methods have been developed to obtain sub-pixel accuracy. The principle is to apply an algorithm across the pixels such that the correlation co-efficient surface (at any given point) is described by a function. Finding the maximum of the correlation co-efficient surface allows one to infer displacement to sub-pixel values.

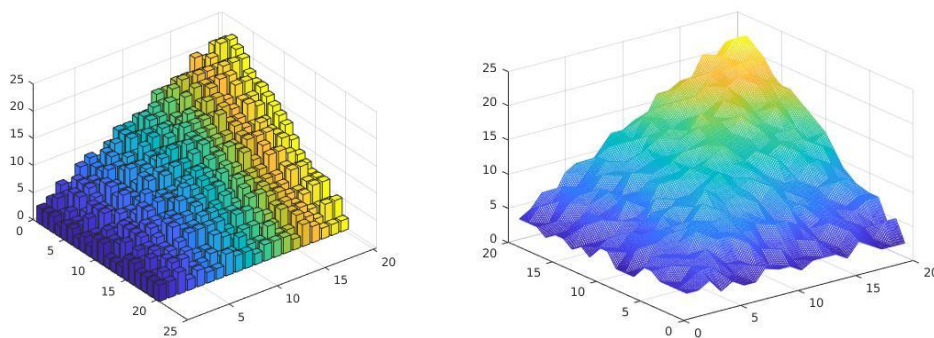


Figure 11: Pixelated and Linear Interpolated.

There are various ways to interpolate between pixels. The example given in Figure 11 uses a linear interpolation between pixels.

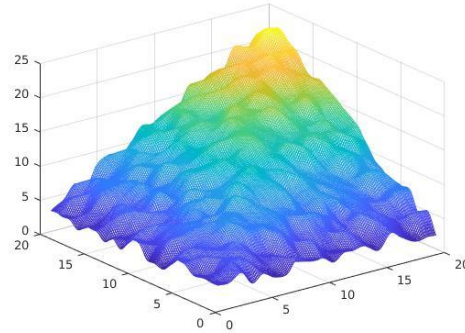


Figure 12: Cubic Interpolation Between Pixels.

As shown in Figure 12, a cubic interpolation gives a different surface on which to maximise the correlation. The choice of which interpolation to use is based on the data presented as well as factors such as the computational availability. A higher-order interpolation scheme has been shown to reduce errors but does increase processing time. Moving from simple linear interpolation to cubic results in large gains in accuracy with diminishing returns for quintic interpolation and higher-order functions [17].

Since the space between pixels is now described by a function the question becomes how to find the position that describes the displacement precisely. One could search the entire space for an extremum, but this seems computationally expensive. The various options and trade-offs are discussed in section 2.2.3.

2.2.2. Cross Correlation Criteria

The cross correlation is executed by running a smaller sub-image of $(2N+1)^2$ pixels, taken from a reference time point, across the larger sub-image of $(2M+1)^2$ pixels, taken at a later time point when the target has moved or deformed. This results in a value that denotes the degree of similarity at each point. This correlation coefficient distribution map is analysed and the point at which it is at an extremum is that point at which they are the “same” [18]. In Figure 13 the subset chosen is run across the image using a standard cross correlation algorithm. This results in the graph shown in Figure 14.

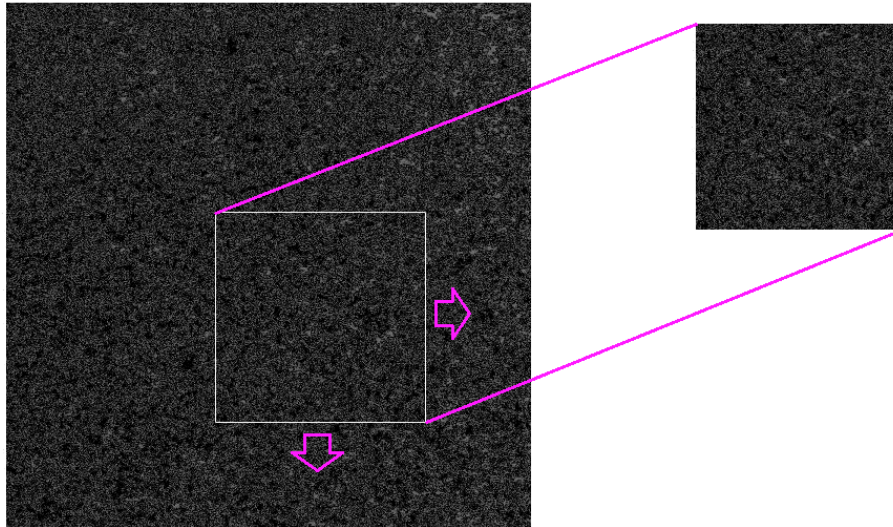


Figure 13: Cross Correlation Process on an Image.

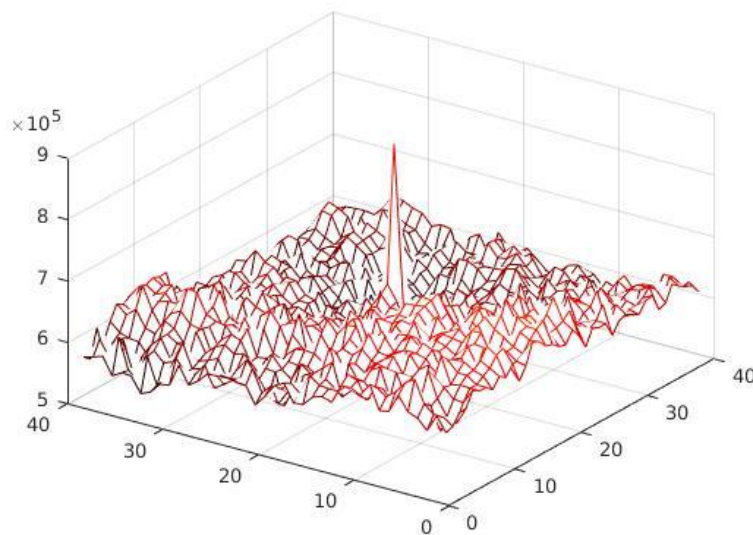


Figure 14: Output of Cross Correlation Process.

There is a clear peak at the point at which the two images correlate. This is where the reference sub-image has been displaced to in the subsequent larger image.

The description up to now has described the tracking of a reference image across a subsequent image to find the area of the highest correlation. In DIC a region of interest (ROI) is chosen on a sample, which is the area where the deformation is going to be monitored. This ROI is then broken into many subsets or facets that will be tracked between consecutive images. In Figure 15 the ROI is given by the large yellow block. The breakdown of this block is the subset/facet allocation. In some applications the subsets may also overlap as long as this still creates the uniqueness per subset.

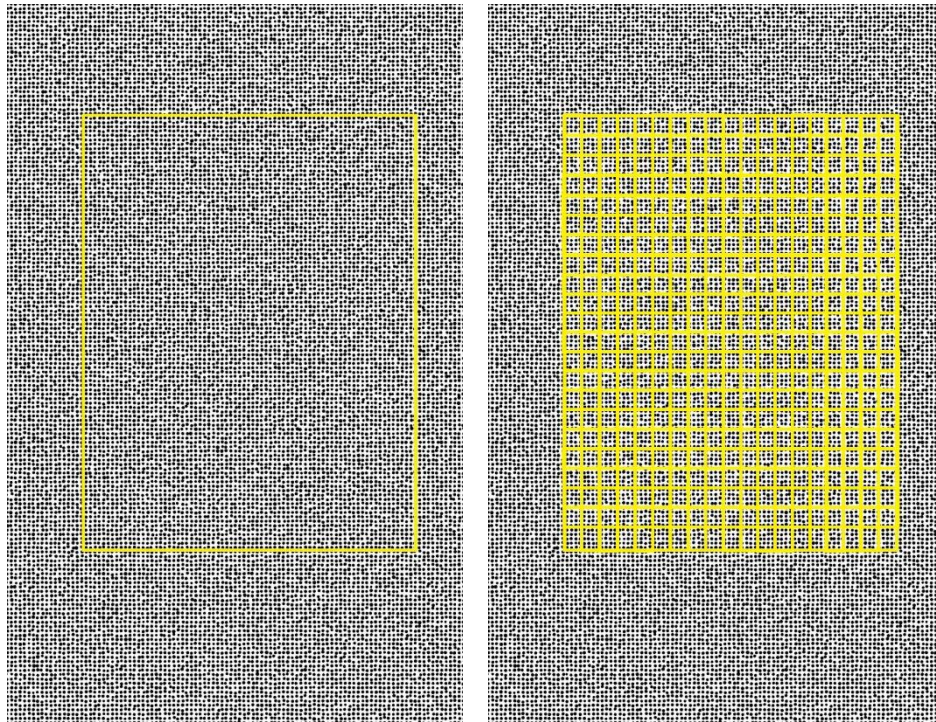


Figure 15: Region of Interest and Subset Allocation.

Each of these subsets is then correlated and tracked in each image of the DIC set. In this way, the deformation of the ROI may be monitored with high spatial resolution, throughout the experiment.

There are many definitions of the matching functions, each slightly more applicable or useful in certain scenarios [19]. The types of correlation criteria can be broken into four main categories namely: Cross Correlation (CC), Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD) and Parametric Sum of Squared Differences (PSSD). The output of most of these criteria can be shown to offer similar mathematical efficacy, but some offer performance benefits, while others show better performance when the intensity level of the pixels change. A robust comparison of all of these was carried out by Pan et al. [7].

Table 1 contains the most common cross correlation equations in the literature [19].

Algorithm	Equation
Cross Correlation (CC)	$C(x, y, x^*, y^*) = \sum F(x, y)G(x^*, y^*)$
Normalised Cross Correlation (NCC)	$C(x, y, x^*, y^*) = \frac{\sum F(x, y)G(x^*, y^*)}{\sqrt{\sum F(x, y)^2 \sum G(x^*, y^*)^2}}$
Zero Normalised Cross Correlation (ZNCC)	$C(x, y, x^*, y^*) = \frac{\sum [F(x, y) - F_m] \times [G(x^*, y^*) - G_m]}{\sqrt{\sum [F(x, y) - F_m]^2 \sum [G(x^*, y^*) - G_m]^2}}$
Sum of Squared Differences (SSD)	$C(x, y, x^*, y^*) = \sum [F(x, y) - G(x^*, y^*)]^2$
Normalised Sum of Squared Differences (NSSD)	$C(x, y, x^*, y^*) = \sum \left[\frac{F(x, y)}{\sum F(x, y)^2} - \frac{G(x^*, y^*)}{\sum G(x^*, y^*)^2} \right]^2$
Zero Normalised Sum of Squared Differences (ZNSSD)	$C(x, y, x^*, y^*) = \sum \left[\frac{F(x, y) - F_m}{\sum [F(x, y) - F_m]^2} - \frac{G(x^*, y^*) - G_m}{\sum [G(x^*, y^*) - G_m]^2} \right]^2$

Table 1: Correlation Equations.

In Table 1 F_m and G_m have been referred to in various equations. These common functions are described by:

$$F_m = \frac{1}{(2M + 1)^2} \sum F(x, y)$$

$$G_m = \frac{1}{(2M + 1)^2} \sum G(x^*, y^*)$$

Equation 2: Pixel Summation Formula.

For the same information presented in Figure 14, the Normalised Cross Correlation would give an output represented by Figure 16. There is very little further information presented in this different approach, but the output is normalised which can be useful when trying to interpret the results.

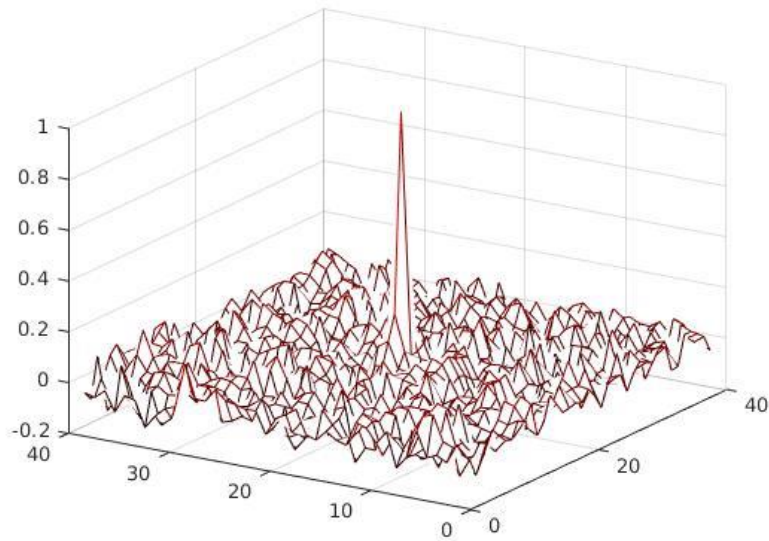


Figure 16: Normalised Cross Correlation Results.

2.2.3. Optimisation Algorithm

Cross correlation is an excellent method for getting the displacement of a deformation to within a few pixels. The final position x^* , y^* relates to the initial x , y by the equations:

$$x^* = x + U_x$$

$$y^* = y + U_y$$

Equation 3: Translation Without Deformation.

where U_x and U_y are the translation of the sample in the x and y directions.

For a sample that undergoes pure translation with no deformation, this method can get an answer to within one pixel. In general, however, this is not the case.

Consider the example including translation and deformation, shown in Figure 17.

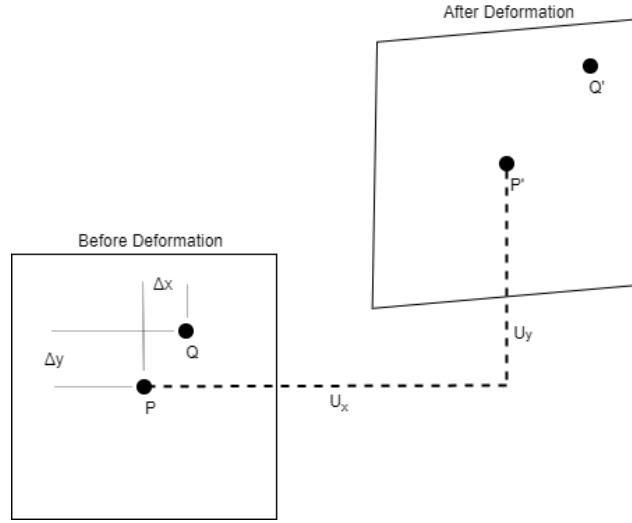


Figure 17: Subset deformation.

The translation of the sample is denoted by U_x and U_y . Given this displacement the position of Q' is now denoted by the equations [12]

$$x^* = x + U_x + \frac{\partial U_x}{\partial x} \Delta x + \frac{\partial U_x}{\partial y} \Delta y$$

$$y^* = y + U_y + \frac{\partial U_y}{\partial x} \Delta x + \frac{\partial U_y}{\partial y} \Delta y$$

Equation 4: Translation with Deformation.

where U_x and U_y denote the displacement at the centre of the subset, the partial derivatives (e.g. $\frac{\partial U_x}{\partial x}$) denote the deformation and Δx and Δy denote the distance to an arbitrary point. The correlation metric, through whichever method is chosen, attempts to find the values for U_x , U_y , $\frac{\partial U_x}{\partial x}$, $\frac{\partial U_x}{\partial y}$, $\frac{\partial U_y}{\partial x}$, $\frac{\partial U_y}{\partial y}$ such that an extremum is found. A coarse/fine method could be chosen, where a coarse search is carried out and subsequently the extremum is honed in on with a fine search, but this can prove to require a large number of calculations.

Some methods have been suggested for improving the efficiency of calculation after the initial guess provided by coarse cross correlation. As mentioned in section 2.2.1, the surface made by the pixels is described by a curve which allows for subpixel registration. To use numerical methods to improve efficiencies this function becomes important. An example of bilinear interpolation is

$$G(x^*, y^*) = a_{00} + a_{10}(x') + a_{01}(y') + a_{11}(x')(y'),$$

Equation 5: Bi-linear Interpolation Equation.

where:

$$a_{00} = G(i,j)$$

$$a_{10} = G(i + l,j) - a_{00}$$

$$a_{01} = G(i,j + l) - a_{00}$$

$$a_{11} = G(i + l, j + l) - a_{00} - a_{10} - a_{01}$$

with the following pixel map representing the input.

$G(i-1,j+2)$	$G(i,j+2)$	$G(i+1,j+2)$	$G(i+2,j+2)$
$G(i-1,j+1)$	$G(i,j+1)$	$G(i+1,j+1)$	$G(i+2,j+1)$
$G(i-1,j)$	$G(i,j)$	$G(i+1,j)$	$G(i+2,j)$
$G(i-1,j-1)$	$G(i,j-1)$	$G(i+1,j-1)$	$G(i+2,j-1)$

Figure 18: Bilinear Interpolation Pixel Map.

One could use the higher order interpolation schemes and the arithmetic would become more complicated as the orders increase, but the basic principle would remain the same.

Since we now have a function defining an arbitrary position in our pixel field, we can use a numerical method to optimise the calculation process. The Newton-Raphson method, first suggested for use in DIC by Bruck et al [4], is the de-facto standard introduced in the literature and this will be covered here.

The Newton-Raphson method is a means of finding successively better approximations to the roots of a function. To use Newton-Raphson we are trying to find the point at which the gradient of the correlation function is a minimum. From Equation 1, the gradient function is as follows:

$$\nabla C(u_x, u_y, \partial u_x / \partial x, \partial u_x / \partial y, \partial u_y / \partial x, \partial u_y / \partial y).$$

Equation 6: Gradient of Correlation Function.

When Newton Raphson is applied to Equation 6 we get

$$\nabla \nabla C(U_k)(U_{k+1} - U_k) = -\nabla C(U_k),$$

Equation 7: Newton-Raphson Correction Equation.

where $\nabla C(U_k)$ represents the Jacobian matrix (J) and $\nabla \nabla C(U_k)$ is the Hessian matrix(H) [4]:

- Each term in the Jacobian matrix is the derivative of the correlation function at guess k.
- Each of the terms in the Hessian matrix are the second derivatives of the correlation function at iteration k.

This equation is often represented in the literature as

$$U_{k+1} = U_k - \nabla \nabla C(U_k)^{-1} \nabla C(U_k)$$

or

$$U_{k+1} = U_k - H^{-1}(U_k) \cdot J(U_k).$$

Equation 8: Rewriting of Newton-Raphson Equation.

The partial corrections are added to the initial guess and the process will continue until convergence is obtained. The arithmetic associated with this maths can get bulky and as such has been left out of this section. All the arithmetic is covered in detail in many papers [12], [4], [19].

The process as described above is an iterative process of finding an optimised solution to a non-linear multivariate problem. There are various numerical analysis methodologies that have been tried in solving this problem. Vendroux and Knauss [20], for instance, introduced a further improvement by suggesting that the Hessian matrix could be adequately approximated for situations with sufficiently small out of plane displacements.

Other proposals have included the Levenberg-Marquart algorithm [17] and the quasi Newton Raphson algorithm [21]. These have both been introduced as means of improving speed efficiencies in the DIC process.

More recently Pan et al. adapted the Inverse Compositional Gauss-Newton method for use with DIC. This allows for a constant Hessian matrix which reduces calculations dramatically [22]. This method shows clear benefits and has become the standard algorithm to use.

2.3. Chronology of Developments

The large number of papers published in this field over the last 40 years can be broken into a foundation-laying phase for the first half and an exploration phase for the second half. The foundation-laying phase defined the mathematical framework, hardware fundamentals, best practice, and accuracy trade-offs for the field. The second phase involved experimentation with varying ideas as well as broadening the field outside of solid mechanics to areas like biological specimens, geological structures, and others [13]. What follows is a brief chronology of the field and the papers that created the highlights in it.

Year	Main Paper	Summary of Content
1982	Digital Imaging Techniques In Experimental Stress Analysis [23]	Uses a laser speckle pattern and camera system to measure distortion and surface displacements.
1983	Determination of displacements using an improved digital correlation method [3]	Introduced the use of a painted speckle pattern over a laser speckle pattern. Use of least squares correlation. Use of iterative coarse fine algorithm. Use of bilinear interpolation.
1985	Applications of digital-image-correlation techniques to experimental mechanics [2]	Uses normalised cross correlation. Uses polynomial interpolation instead of bilinear. Uses a "two parameter" iterative algorithm.
1986	Application of an optimized digital correlation method to planar deformation analysis [5]	Introduces the use of the Newton-Raphson algorithm. Results in a 20x increase in speed with the same accuracy.
1989	Digital image correlation using Newton-Raphson method of partial differential correction [4]	Expanded on Newton-Raphson method. Uses the bicubic spline interpolation for sub-pixel registration.
1993	Accurate measurement of three-dimensional deformations in deformable and rigid bodies using computer vision [24]	First implementation of 3D DIC.
1993	Digital speckle-displacement measurement using a complex spectrum method [25]	Used FFT for speckle displacement measurement. Results are good but are shown to not deal well with large rotation/deformations.

1998	Submicron deformation field measurements: Part 2. Improved digital image correlation [20]	Introduced an approximation for the Hessian Matrix that improved calculation complexity. This overcame a huge milestone for wide spread adoption of Newton-Raphson as an algorithm in this field.
1999	DVC - three dimensional strain mapping using xray tomography [26]	Introduction of Digital Volume Correlation.
2001	Equivalence and efficiency of image alignment algorithms [27]	Introduces Inverse-Compositional Gauss-Newton method.
2005	An Evaluation of Digital Image Correlation Criteria for Strain Mapping Applications [28]	Compares various correlation criteria on worst case scenario images.
2010	Equivalence of digital image correlation criteria for pattern matching [7]	Compares correlation criteria: <ul style="list-style-type: none"> • ZNCC • ZNSSD • PSSD
2012	Optimization of a three-dimensional digital image correlation system for deformation measurements in extreme environments [29]	Introduced blue light sources as a means of getting accurate results in high temperature environments.
2017	3D shape, deformation, and vibration measurements using infrared Kinect sensors and digital image correlation [30]	Exhaustive technical discussion of 3D DIC. Uses IR light for measurements and compares these to standard DIC.

Table 2: Chronology of DIC Developments.

2.4. 3D DIC Theory

The discussion thus far has focussed on a single camera measuring deformation in a single plane. The goal of this project, however, is to measure deformation in three-dimensional space. Although various attempts have been made to achieve this with a single camera [31]–[33], the focus of this project will be the process of achieving this with two cameras.

The setup for any experiment that involves 3D point location, aside from those single camera options mentioned, is as shown in Figure 19.

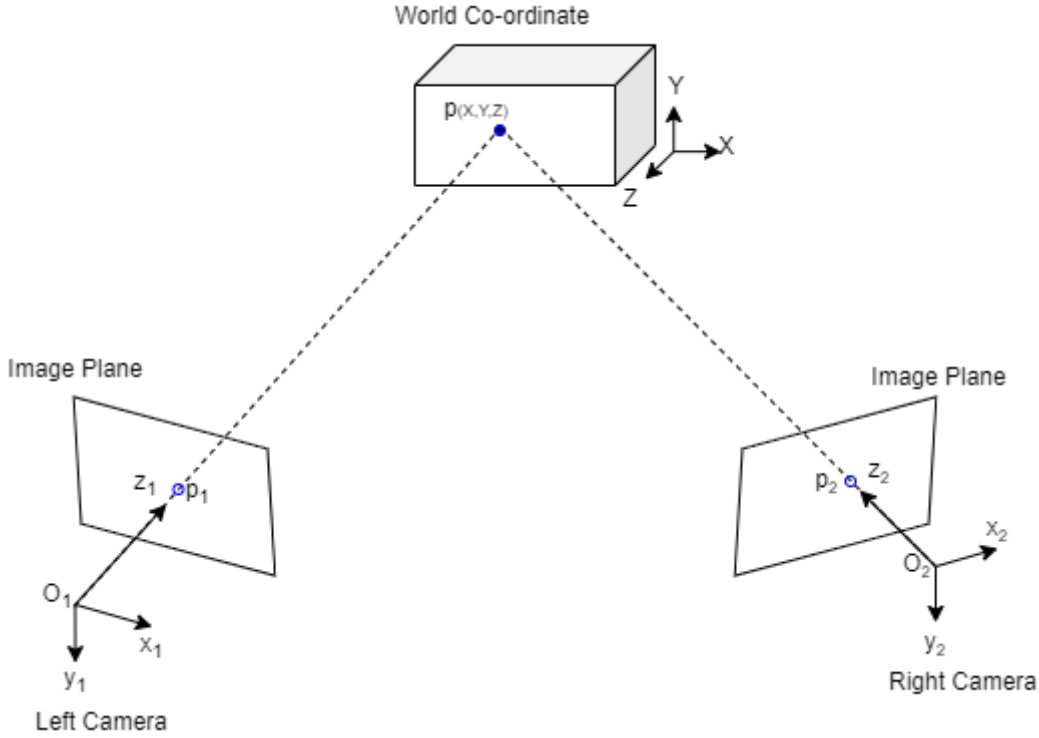


Figure 19: 3D-DIC Experimental Setup.

Retrieving the 3D information about the point p with relation to the points p_1 and p_2 in the setup above involves 2 steps:

1. Calibrate the cameras to get the camera parameters (the geometry of the stereo vision system).
2. Perform matching to find the same physical points in the two images [30].

2.4.1. Camera Calibration

The goal of camera calibration is to express the point p_1 (Figure 19) in relation to point p . This means we need to express an arbitrary point (x_1, y_1, z_1) as a function of the real world point (x, y, z) . The equation to achieve this is

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [R \quad T] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix},$$

Equation 9: Real World to Image Plane Conversion (Extrinsic Parameters).

where R and T indicate the rotation and translation from the real coordinates to the camera coordinate system, respectively. These are known as the extrinsic parameters of the camera. Extrinsic parameters are the parameters external to the camera. These parameters are the rotation and translation from the world frame used (i.e., the fixed coordinate system used in the “world”).

The camera model is further calibrated by the extraction of the intrinsic parameters. The intrinsic parameters are those parameters internal to the camera/sensor (i.e., focal length, skew, geometric distortion etc.). The intrinsic parameters allow for a mapping of the camera reference frame to the pixel location:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \frac{1}{z_1} \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix},$$

Equation 10: Camera Co-ordinates to Pixel Co-ordinates (Intrinsic Parameters).

where α and β are the focal length in pixel units, γ is a skew factor and (u_0, v_0) are the coordinates of the principal point.

Ultimately the final translation from real world co-ordinates to pixel location is captured in the formula below [34]:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \frac{1}{z_1} \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Equation 11: Real World to Pixel Location Formula.

Equation 11 does not explicitly cater for lens distortion. The lens distortion can be accommodated by introducing coefficients for radial, prism and tangential distortion. In order to cater for the lens distortion, the output of Equation 9 needs to be adjusted to cater for the distortions mentioned:

$$\widetilde{x}_1 = (1 + k_0 r^2 + k_1 r^4 + k_2 r^6 + k_3 r^8 + k_4 r^{10})x_1 + (k_5 + k_7 r^2)r^2 + (k_9 + k_{11} r^2)(r^2 + 2x_1^2)$$

$$\widetilde{y}_1 = (1 + k_0 r^2 + k_1 r^4 + k_2 r^6 + k_3 r^8 + k_4 r^{10})y_1 + (k_6 + k_8 r^2)r^2 + (k_{10} + k_{12} r^2)(r^2 + 2y_1^2)$$

$$r^2 = x_1^2 + y_1^2.$$

Equation 12: Lens Distortion Corrections.

Here (k_0, \dots, k_4) represents the radial distortions coefficients, (k_5, \dots, k_8) the prism distortions coefficients and (k_9, \dots, k_{12}) the tangential distortions coefficients. In this equation (x_1, y_1) represents the distortion free pixel location and $(\widetilde{x}_1, \widetilde{y}_1)$ is the distorted point.

To obtain all the parameters that calibrate the camera the following process needs to be followed [35], [36]:

- Let the cameras observe a planar pattern (usually a checkerboard, or grid of circles) at a few different orientations.
- Detect the feature points of the planar image.
- Estimate the 5 intrinsic parameters and all extrinsic parameters of the system.
- Estimate the coefficients of the distortion model.
- Refine all parameters by optimising until the calibration points are predicted.

The mathematics of this process will not be covered here but has been covered extensively by Zhang [35] and Burger [36] amongst others [30], [34],[37].

2.4.2. 3D Calibration Specifics

The specific case of calibration of a stereo vision system uses most of the logic outlined in section 2.4.1. Calibration in this context involves determining the intrinsic parameters of each camera and the relative position and orientation (extrinsic parameters) between them [38].

Referring again to Figure 19 the main outcome of 3D stereo calibration is to find a mapping between the two cameras such that p_1 and stereo-correspondent p_2 both represent the point $p(X,Y,Z)$ in the real world. To reduce the search space associated with matching the same point in the two cameras an epipolar constraint can be used [39].

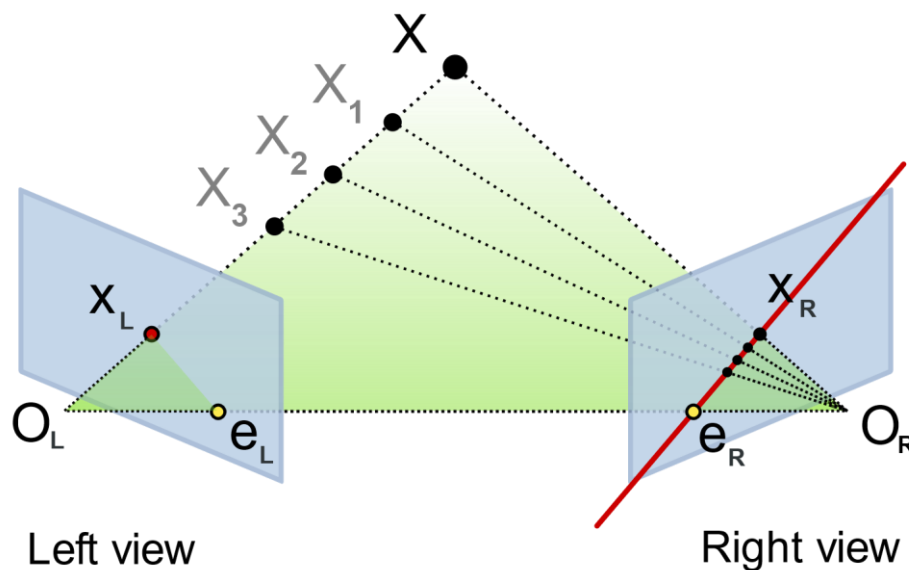


Figure 20: Epipolar Geometry Concept Diagram [40].

Referring to Figure 20, the centre point of each camera (O_L and O_R) projects onto the other at a distinct point, e_L and e_R respectively. These points are said to be epipolar points. The points O_L , O_R , e_L and e_R all lie on one 3D line.

The camera with centre O_R sees O_L-X as a line. This corresponds to the line X_R-e_R on the image plane of the right camera, which is known as an epipolar line.

If the relative position of both cameras is known this leads to an important constraint for stereo vision:

- For a given point in one camera view, the projection of a point on the camera plane must be contained within a known epipolar line in the other camera. This means it is possible to test if two points correspond to the same 3D point.
- If the points X_L and X_R (in Figure 20) are known and it has been confirmed that they represent the same 3D point, this means that X can be calculated by triangulation.

Epipolar geometry forms the basis for 3D stereo calibration where metric information is required [41].

2.4.3. 3D DIC Process

The process for 3D DIC involves four main steps [42]

- Stereo calibration (as described in section 2.4.1 and 2.4.2).
- Stereo and temporal matching.
- 3D shape reconstruction.
- Displacement and strain field measurements.

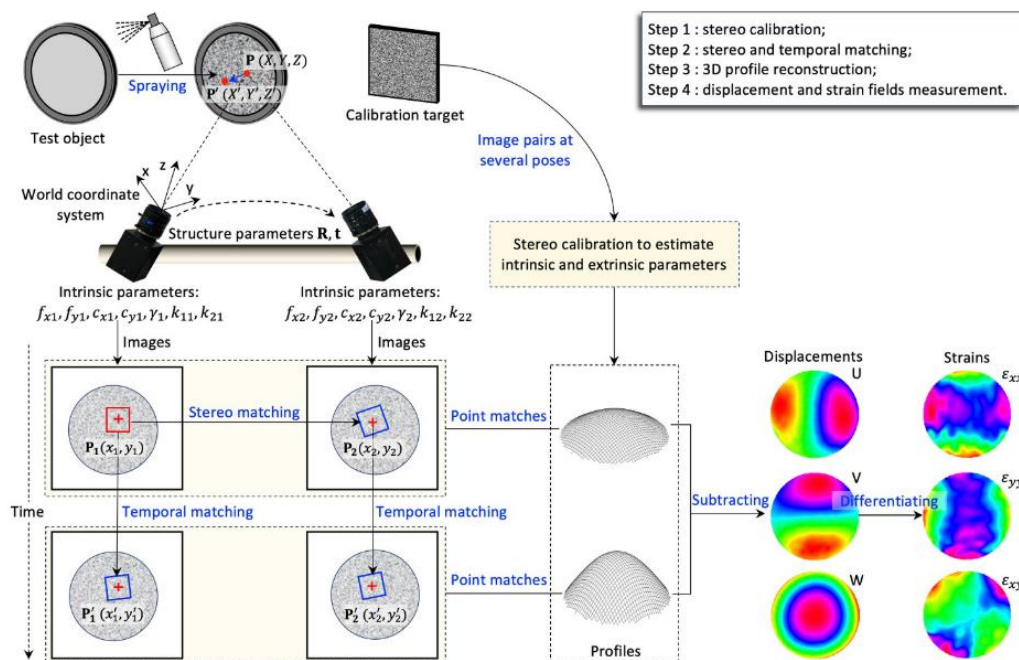


Figure 21: Schematic showing the principles of stereo-DIC [42].

As mentioned, step 1 of the process has been described in section 2.4.1. For a stereo camera system, the outcome of this process is that the intrinsic parameters of each camera as well as the relative position and orientation between them has been determined.

The step of stereo and temporal matching consists of either template matching or feature matching between images, and subsequently template matching temporally (as would be the case for 2D DIC) [38].

The two cameras in the stereo vision system capture images of the same target. The problem now is to find the same points in each image. To minimise the search area, or to give a good initial guess of the location of a left camera subset in the right camera's frame, various techniques can be used. One such technique is to use the epipolar constraint described in section 2.4.2. This technique is commonly used since it requires no further implementation of algorithms. The basic premise of this approach is to use the epipolar constraint to define the line segment along which to search and to use the standard correlation criteria already implemented to find the stereo-correspondent point.

Another way is to use feature matching algorithms from general computer vision. These include BFM (brute force matching), FLANN (fast library for approximate nearest neighbours) matching, SIFT (scale invariant feature transform) or SURF (speeded up robust features) amongst others [30],[43]. The work done in this project used a basic brute force matcher. The papers referenced suggest that SURF and SIFT would work well but given how these algorithms work their efficacy for the fine detail of a speckle pattern is questionable.

Running these algorithms finds similar features in two images and matches them. This gives a good indication of how the two images differ from each other in space and as such gives a good idea of where, a given subset in one frame might be in the other. Once again, on finding a good initial guess, the standard correlation criteria are used to find the stereo-correspondent points.

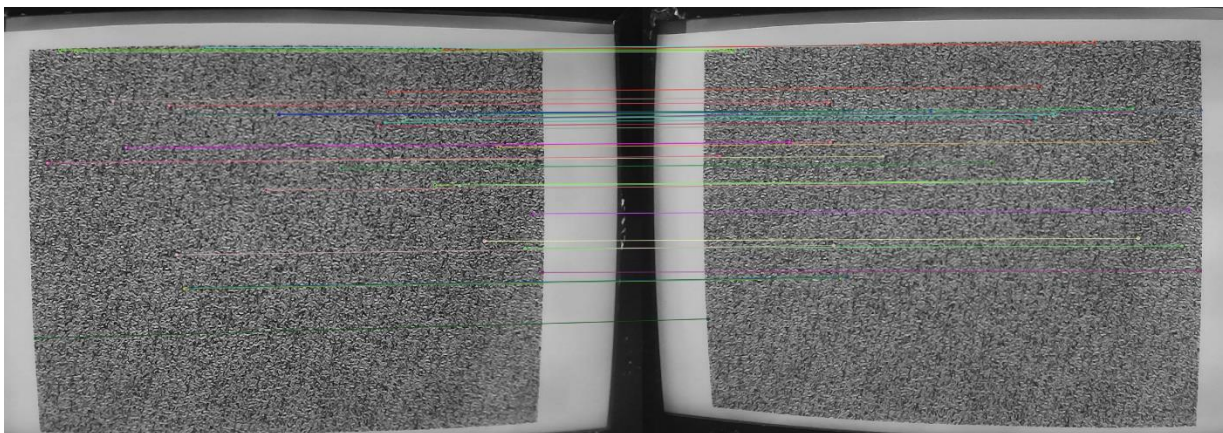


Figure 22: Feature matching between two frames.

From this point the standard 2D DIC algorithms for correlation criteria and interpolation will apply.

The output of this process is that (referring to Figure 19) we now know the positions of p_1 , p_2 and P within the co-ordinate system created by the calibration process. The 3D reconstruction step is therefore reduced to a triangulation problem. Conceptually, triangulation is simple, but implementing triangulation has complications. Principally, the complications are the effect of digital noise in the sampling of the data as well as the error introduced from compromises in accuracy. These issues make it such that the actual lines associated with our three points may not coincide but exist in a region of variance. How to handle this is beyond the scope of the discussion here but is covered in some detail by Hartley et al. [44].

This process is then carried out within each frame and between the two frames for each subsequent captured image in the deformation process. The result is a set of displacements in X, Y and Z. Differentiating these displacements will then result in the strain.

2.5. Efficiency Improvements

The fundamental concepts behind DIC are now almost 40 years old. As stated in the introduction, in that time computer power has changed enormously but the fundamental logic behind DIC has remained consistent. The last 40 years have introduced improvements in speed and accuracy which have involved areas of lighting, target preparation, hardware choices and algorithmic improvements amongst others. For the sake of brevity, only the highlights of these improvements will be covered here. The referenced papers cover these topics in more detail than is required by the scope of this project.

Note that the area of camera calibration has been discussed above, but from a standard practice perspective. There is an entire area of research devoted to various methods of camera calibration and their effects on accuracy which are discussed in detail in [33]–[36].

2.5.1. Improvements in Methods, Preparation and Tools

2.5.1.1. Target Preparation

DIC relies on the target having a variation in intensity on the surface such that a given subset can be differentiated from the rest of the search space easily and after possible deformation in any direction. The characteristics of the speckle pattern are therefore critical to the accuracy of the result.

There are four main characteristics of a good speckle pattern:

- High contrast: the grey levels should vary with large gradients
- Randomness: the pattern should be highly variable and non-periodic
- Isotropic: the pattern should have no directionality
- Stability: the pattern should adhere to the sample surface under deformation without undue change to shape or intensity level.

In a set of papers, Reu discussed the various characteristics of speckle patterns and their fabrication [45]–[48]. These articles covered the size and density of speckles, aliasing and edge sharpness.

A thorough review of speckle patterns and the various methods of application was undertaken by Dong and Pan [8]. This paper also lays out a method of assessment of the quality of a speckle pattern.

Another means of speckle pattern evaluation has been developed by Crammond et al. [49]. This method relies on edge detection within the speckle pattern to determine the quality of the pattern. This approach has the added attraction of being implementable entirely within a machine vision context.

In stereo vision systems a very recent paper has suggested the use of an electronically generated speckle pattern on a tablet [42]. Since a stereo vision system's accuracy is heavily dependent on the accuracy of the calibration, generating the pattern electronically offers the promise of significant accuracy improvements.

2.5.1.2. Lighting

Since the intensity of the grey level plays such a big part in the accuracy of DIC, it follows that the means of illumination would have a similar effect. The importance of illumination can be seen in the experiment done by Yoneyama et al. [50]. In this experiment the deflection of a bridge was measured by DIC. Since the experiment had to be done outside it was carried out at night to control the light source. Another means of controlling the light source was proposed by Pan et al. [51]. This proposes the use of monochromatic light to illuminate the target. A bandpass filter is then applied to the input to the camera. This controls the illumination of the target and completely removes the harmful effect that ambient light can have on measurements.

Vanlanduit et al. also introduced the use of stroboscopic illumination to measure crack propagation [52]. They took advantage of the cyclic nature of the fatigue tests to subsample the data using a stroboscopic light. This allowed for the use of less high-end equipment but with very similar results.

2.5.1.3. Single Camera Solutions (Stereo-DIC)

As has been covered, stereo DIC uses two cameras. These cameras must be synchronised, which can be a complicated exercise. Furthermore, the requirement for two cameras puts a physical constraint on the experiment, which can sometimes cause problems. To this end some work has started appearing regarding the use of single-camera stereo DIC [31].

The basic premise is to expose the same sensor to the image of the sample from two different angles using a lens, diffraction grating [31], biprism [53], or some other form of light path manipulation. An example of the experimental setup is given in Figure 23 below.

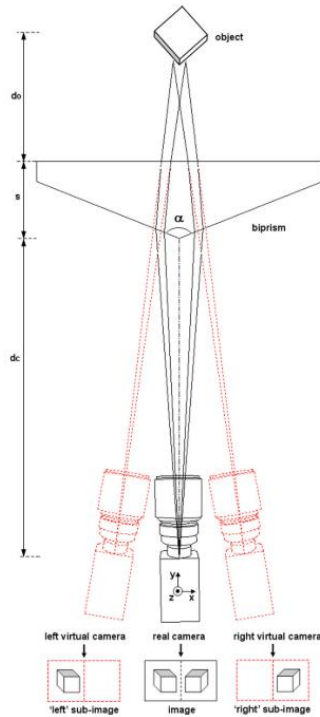


Figure 23: Single Camera Stereo DIC Experimental Setup [54].

Although this method removes the need for camera synchronisation and reduces the requirements for expensive equipment it does come with its own complications. The optical design of such a system is non-trivial and the implications of this are that one loses the flexibility to place the camera/s anywhere, they now have to be at a very specific angle and focal distance from the optical device being employed.

2.5.2. Algorithmic Improvements

2.5.2.1. Correlation

Many of the early improvements in correlation lead to its wide applicability today. This section will not cover the early improvements, since these have been covered already in sections 2.2.2 and 2.3, but will focus on more recent developments in this area.

As mentioned in section 2.2.3, the Newton Raphson method is used for accurate subset matching. This is computationally intensive. The work by Vendroux and Knauss [20] improved the calculations by allowing for the approximation of the Hessian, but this is only applicable in certain situations. In 2012 Pan et al. suggested an alternative, namely the inverse compositional Gauss-Newton (IC-GN) method with the ZNSSD correlation criteria (see Table 1). The main attraction is that the Hessian matrix remains constant and can be calculated beforehand [22].

IC-GN was first introduced as an improvement to the Lucas-Kanade algorithm for image alignment but was quickly adapted to the DIC space [55]. The main benefits of using IC-GN is the reduced computation required, but a further benefit is an improved noise robustness [56].

Various attempts have been made to introduce frequency spectrum comparisons to the correlation space, but there have been challenges with this approach. Applying the Faster

Fourier Transform to Cross Correlation (FFT-CC) leverages the very efficient algorithms for FFTs, which is very attractive from a speed perspective. Figure 24 shows an FFT run on a speckle pattern. The centre of the magnitude spectrum represents low frequency components of the image and the outer edges the high frequencies. The FFT of the reference image is compared to the FFT of the subsequent images to estimate correlation. Since in the frequency space the frequency components that are noisy (i.e., do not represent the underlying structure of the image) can be removed, the comparison does not need some of the complex correlation calculations that have been explained previously. Further once these frequency components are matched the phase shift is used to calculate the actual displacement. All of this results in a comparably fast algorithm [57]. An example of this frequency space representation of a speckle pattern is given in Figure 24 below. The python code to generate the magnitude spectrum can be found in Appendix 5.

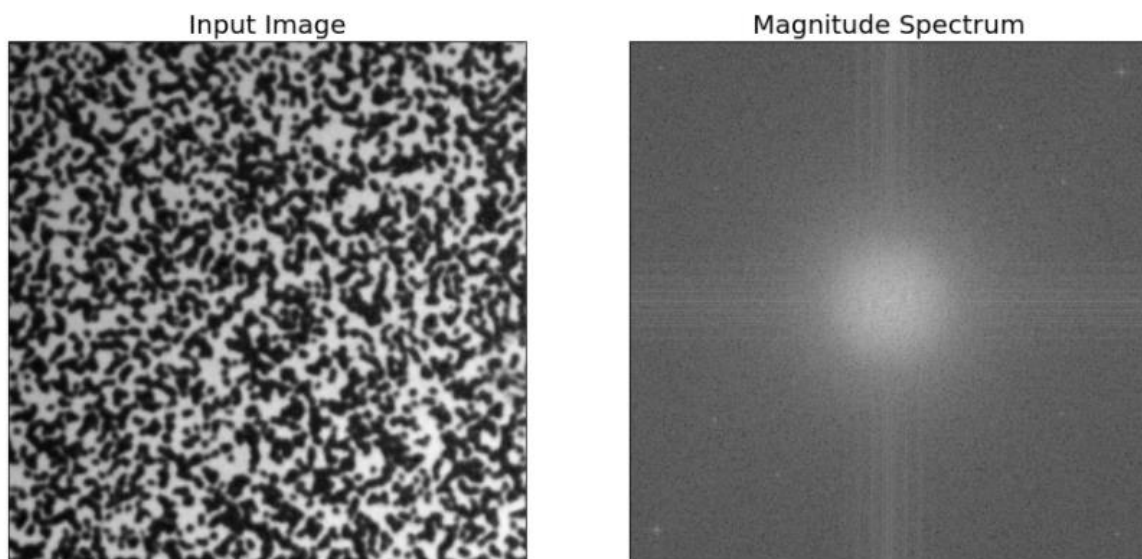


Figure 24: Magnitude Spectrum of Speckle Pattern.

The main drawback of FFT-CC is that it is only effective to samples that undergo translation. Large deformation in the sample means that the FFT of the deformed image would bear very little resemblance to the original FFT. It can however, handle very simple deformation and with further work this could become a very promising area.

2.5.2.2. Interpolation

In various papers bilinear interpolation, bicubic interpolation, bicubic B-spline interpolation, biquintic B-spline interpolation and bicubic spline interpolation are used. Schreier et al. suggest bicubic spline or biquintic spline interpolation [17],[58].

As one would expect, for pixel interpolation the higher the order of the interpolation the better. Running numerical studies on speckle patterns Schreier et al. quantified the error associated with the various types of interpolation. They proved that higher order interpolation is

preferable. Jumping from linear to cubic interpolation garnered the most dramatic improvements with diminishing returns with increasing order after that.

2.5.3. Real Time DIC

In the last few years some work has started emerging that, like this project, investigates the feasibility of real-time 3D DIC measurement. This research largely comes from the biomedical and civil engineering arenas, but there has been some general work into improving efficiencies with real-time operation in mind.

In the biomedical sphere attempts are being made to use DIC as an aid to various procedures. In most of these applications real-time processing has been achieved mainly because sub-pixel accuracy is not important or because only translation (not deformation) of the target needs to be measured [59] [60]. This is because in most medical measurements currently employing this technology, temporal resolution (i.e. timing) is more important than spatial accuracy.

Other real-time work has focused on civil engineering and specifically the real-time monitoring of structures such as bridges. Once again, the real-time processing is achieved because only a few scattered subsets are monitored for accurate deformation [61], with subsets between these being ignored.

To re-iterate, the general DIC process involves two main steps:

- Find the integer pixel point of highest correlation.
- Run a sub-pixel algorithm to find the sub-pixel accurate displacement.

This second step has gone through a lot of optimisation, culminating in the work by Pan et al. that suggested the Hessian matrix could be calculated once and used as a constant [22]. It is currently suggested that the first step, i.e., finding the initial integer pixel displacement is the step taking up the most processing time. Improving this process has therefore become the focus of a lot of work in the real-time space [62].

Some of the suggested methods here include:

- Fast Fourier Transform [25].
- Genetic Algorithms [63].
- Difference Evolution [64].
- Particle Swarm Optimisation [65].

An explanation of these algorithms is beyond this scope of this work, but these are very interesting avenues of research.

It is also worth noting that the real-time work done so far, in most of the papers referenced, has concentrated on 2D DIC. This project attempts to understand the implications of running real-time 3D DIC. These updated concepts used in the 2D space will be implemented in the 3D DIC space going forward and this will add to the efficiencies we discuss going forward.

3. Design Considerations

As covered in the introduction the question being considered is whether real-time DIC can be effectively undertaken using off the shelf, cost effective components, and if so what the effects on accuracy and speed would be. To start investigating this properly it becomes apparent that one would need to build a DIC system in which one can pull the various levers required to test the concepts mentioned.

The first step in this design process is to consider the various hardware platforms available. The single board computer landscape is growing at an exponential rate and choosing one requires navigating a cost benefit analysis across the space [66].

In addition, in order to evaluate the question under consideration a DIC software suite would be required. There are many commercial options available but for this application an insight into the code base and the ability to adjust it would be required. An open-source code base would therefore have to be found and evaluated.

3.1. Hardware Design

The investigation of the effect of the increase in computing power on the DIC landscape, is the fundamental concept under investigation. We set out to understand what is possible in real-time 3D DIC given the vast improvement in computing power in the last 40 years. The other point of interest was whether this could be done in a cost-effective way.

An option would be to run the chosen software on an extremely powerful desktop computer. This would answer the question in part, but the really interesting investigation is whether one could, given the advances in computing and specifically in single board computing, break the shackles of the traditional computer and effectively run real-time 3D DIC on a standalone piece of hardware.

Large desktop computers are also less desirable given that the ultimate application of this investigation would be bulge tests. For bulge tests on biological materials, performing the test at the clinical site where the material is harvested is very desirable. As a result, portability is a highly desired requirement of the final hardware design.

Laptops would partially satisfy the portability requirement but in most instances they do not allow for readily available peripheral support to capture multiple video streams simultaneously. The single board computer landscape does not (necessarily) limit the peripheral capture in this way.

To answer this question the first stumbling block would be choosing from the huge number of single board computers available. A value system needed to be defined to choose the correct option for this application.

The main considerations within the value system were (in order of importance):

- Operating system support – the ability to handle an operating system mainstream enough to compile and run the chosen software
- Peripheral support
- Processing power/available resources
- Cost
- Extensibility – the ability to consider various hardware options in the experiments.

Various options were considered but for brevity's sake only the short list is included in Table 3.






Device	Estimated Price* (*mid 2021) All prices rounded to the closest R100	Image
<p>Raspberry Pi (3/4)[67]</p> <p>✓ <i>Cheap, ubiquitous, well supported</i></p>	<p>R1500</p>	
<p>Beagle Bone Black[68]</p> <p>✓ <i>Cheap, well supported</i></p> <p>✗ <i>Limited peripheral support onboard</i></p>	<p>R1300</p>	
<p>Jetson AGX Xavier Developer Kit[69]</p> <p>✓ <i>Well supported, high powered processor, GPU (CUDA support), good peripheral options</i></p> <p>✗ <i>Very Expensive</i></p>	<p>R17000</p>	
<p>Jetson Nano Developer Kit[70]</p> <p>✓ <i>Cheap, well supported, high powered processor, GPU (CUDA support)</i></p>	<p>R1200</p>	
<p>Asus Tinker Board[71]</p> <p>✓ <i>Cheap, well supported</i></p> <p>✗ <i>Availability issues, chipset is new (concerns about library support)</i></p>	<p>R1500</p>	

Table 3: Single Board Computers Evaluated.

Ultimately the Jetson Nano Developer Kit was chosen. The Raspberry Pi (especially with the release of the Raspberry Pi 4 in 2019/2020 [72]) was very seriously considered but the Jetson Nano was eventually chosen since it includes the ability to run parallel processing functions across its CUDA² cores. Since DIC is (in theory) an extremely parallelisable process, this feature has the potential to yield interesting results in the future. The use of CUDA cores is not without precedent which lends further credence to the idea of choosing this as the hardware platform [73]–[76].

The one drawback of the Jetson Nano over the Jetson Xavier is the fact that the Nano has only one MIPI CSI-2 DPHY lane. This means that only one CSI camera can be connected to the Nano³. Since two cameras are clearly needed it was decided initially to use matching USB webcams. Although this worked, USB webcams are (by design of the underlying USB infrastructure) unsynchronised cameras. Initially the idea was that this would have to be characterised however during the progress of this project a cost-effective module came to market that allowed for synchronised camera sampling on the Jetson Nano. The Arducam 12MP Synchronised Stereo camera kit allows for synchronised sampling of two video streams to be delivered through the single CSI-2 DPHY lane on the Jetson Nano. This is discussed further in section 3.4.2.

² Compute Unified Device Architecture is a proprietary NVIDIA technology for efficient parallel computing.

³ The new version of the Jetson Nano does have 2 MIPI CSI-2 DPHY lanes but at the time of purchase this was not the case.

3.2. Software Design

The software design was the main part of the build for this project. The requirements for the software were tied into the main goals for the project. The high-level requirements are listed here:

- The software shall be written to optimise speed of execution.
- The software shall be able to, in real time, execute 3D DIC.
- The software shall be able to report, in real time, aspects of the 3D DIC while it is executing.
- The software shall generate data that can be used for post processing of the 3D DIC after the test is complete.
- The software shall only use libraries for which the source code is available.
- The software shall be portable across various hardware platforms.

With these requirements in mind the following design decisions were made:

- The software will be entirely written in C/C++:
 - Implementation in Matlab would not be possible on the chosen hardware.
 - Implementation in Python is, in general, a slower execution environment to C/C++. Although Python can encapsulate C++ layers, which can make it comparable speed-wise, the evaluation of whether the use of Python would be speed comparable was not a question that needed to be answered.
 - Using various languages for various parts of the process would not be speed efficient and would also cause issues due to the real-time nature of the experiment.
- The chosen software libraries will need the least number of dependencies possible to reduce development complications.
- The software will generate data (for postprocessing) that will be compatible with standard software packages used for this purpose (e.g. ParaView)
- The software will use the CMake build platform for portability.

3.2.1. Open-Source Software Choices

There are many options available for DIC in the commercial, proprietary software space. Some examples are:

- Dantec Dynamics.
- GOM Correlate (a Zeiss company).
- Vic-Software (an isi-sys product).
- MatchID.

MatchID provides various prebuilt software kernels that one uses to develop a higher-level application. However, as the source code is not available, this is of limited usefulness in this application. The other examples provide no access to individual blocks of functionality and are geared towards traditional post-processing analysis. They are therefore unsuitable for this investigation.

A review of the open-source space showed that there are established DIC libraries available. This has meant that high quality software for use in these complicated algorithms has become available to the public with very permissible licenses [77]. In this space, three main contenders stood out:

- NCORR: <http://www.ncorr.com/>
 - Developed at the Georgia Institute of Technology.
 - Developed as part of a master's degree to gain a better understanding of the algorithms.
 - Developed in Matlab but has a C++ port.
 - Focuses on 2D DIC.
- DICE: <http://dicengine.github.io/dice/>
 - Developed at Sandia National Laboratories.
 - Produced in the Center for Computing Research.
 - Developed completely in C++.
 - Allows for great flexibility.
- pydic: <https://gitlab.com/damien.andre/pydic>
 - Developed at the University of Limoges (France).
 - Written in Python.
 - Affiliation to the University is unclear.

The last of these is immediately discounted due to the language of choice (Python) but it has been included here as a reference since in any other application this is an incredibly useful resource.

NCORR and DICE are very similar in that the fundamental software core is written in C++, the software is widely accepted to work well, and the code is of good quality. Ultimately, DICE was chosen for this project, for the following reasons:

- The fundamental structure of the code was easier to follow.
- The dependencies required to compile the code were easier to resolve.
- The software used libraries the researcher was more familiar with.
- Flexibility of the correlation process without code changes.
- Sandia National Laboratories has a recognised track record in this area.

3.2.1.1. Digital Image Correlation Engine

DICE is an open-source piece of software developed by Sandia National Laboratories [78]. It is capable of computing full field displacement and strains from a sequence of images or a video file. Since the engine is written in C++ the software is cross platform with some very good instructions for building on Linux.

The very high-level architecture of DICe is given below:

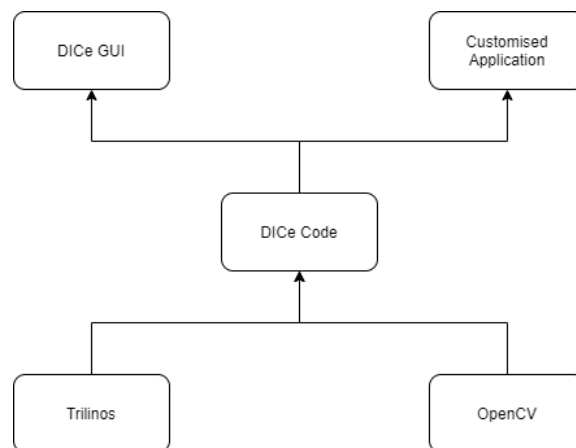


Figure 25: DICe Software Architecture.

[Trilinos](#) is an open-source project containing software packages for the execution of large scale, complex multi-physics and engineering problems. It contains structures to store data and allows for complex mathematical operations on that data such as linear and non-linear solvers.

[OpenCV](#) is an open-source computer vision and machine learning software library. The software contains over 2000 algorithms including image correlation algorithms and feature extraction and matching algorithms.

To allow for complete cross platform building the Trilinos and OpenCV packages need to be built on the required platform. DICe is then built on top of those. One can then link in the DICe libraries to create a custom application. This is the approach that has been taken here.

The DICe GUI is an [Electron](#) application that calls pre-built binaries created during the DICe compile process. This frontend allows for the use of this library as a standalone tool for general digital image correlation requirements.

3.3. Integration and Build

As discussed, the design decisions so far have resulted in using the

- **Jetson Nano Developer Kit.**
- **Digital Image Correlation Engine (DICE).**
- **Arducam 12MP Synchronised Stereo camera kit.**

What follows is a description of the process of integrating these aspects into a useable platform to allow for the evaluation of the underlying question of this project.

3.3.1. Code Development

The instructions to build the DICE software are laid out in intricate detail with many helper scripts for the Linux distribution Ubuntu 16.04. To reduce the initial risk of unknowns the first pass at building the software framework for this project was conducted on this precise platform. Using [Oracle VM Virtualbox](#), a virtual machine with the required operating system was created. The software was built in this environment and all the complications of building the software were discovered and handled during this process.

Once the mechanics of building the framework on the recommended operating system were understood the challenge of building it on the Jetson Nano could be undertaken. The Jetson Nano comes with Ubuntu 18.04 as the operating system. The problems encountered here included:

- the fact that system libraries have been updated, some removed, some renamed and later versions of build tools are standard.
- the fact that the version of Trilinos supported did not cater for non amd64 based architecture (The Jetson Nano runs ARM architecture).

The problem of the supported architecture was handled in a subsequent release of the Trilinos library. The official documentation suggests the use of version 12.4.2, however for the Jetson Nano version 12.14.1 had to be used. Thankfully the API was backwards compatible and no further changes needed to be made to the DICE code with this change.

Ubuntu 18.04 also changed the location of the BLAS and LAPACK libraries and as such the CMake files had to be updated to reflect this.

By far the most complicated problem to resolve was the fact that the linking of the library failed due to the order in which the libraries were included. This problem did not manifest in Ubuntu 16.04 but became apparent in 18.04. The details are beyond the scope of this document, but this involved a low-level investigation into how the build of DICE happens. Once the error was identified, the author fed back the correction to the developers of DICE, who have now included this in the more recent official releases of DICE [79].

3.3.2. Experimental Platform

To test the concepts discussed to this point an experimental setup was required that could do two main things. It needed to have a controlled motor and hence be able to create a known displacement in X, Y or Z and ultimately a known 3D deformation. It also needed to be able to do this at a speed that could be controlled.

In the original project plan this test jig would have been provided, but due to the unique logistics around the coronavirus pandemic in 2020 this could not be achieved. To work around this a test jig was created as part of this project.

The experimental setup was required to simply move in one axis. By placing the cameras in various positions one could then run experiments that track X, Y or Z displacement. The majority of the tests would measure 3D displacement in the Z axis but measuring displacement in the other axes should be possible. The proposed setups for these are shown in Figure 26.

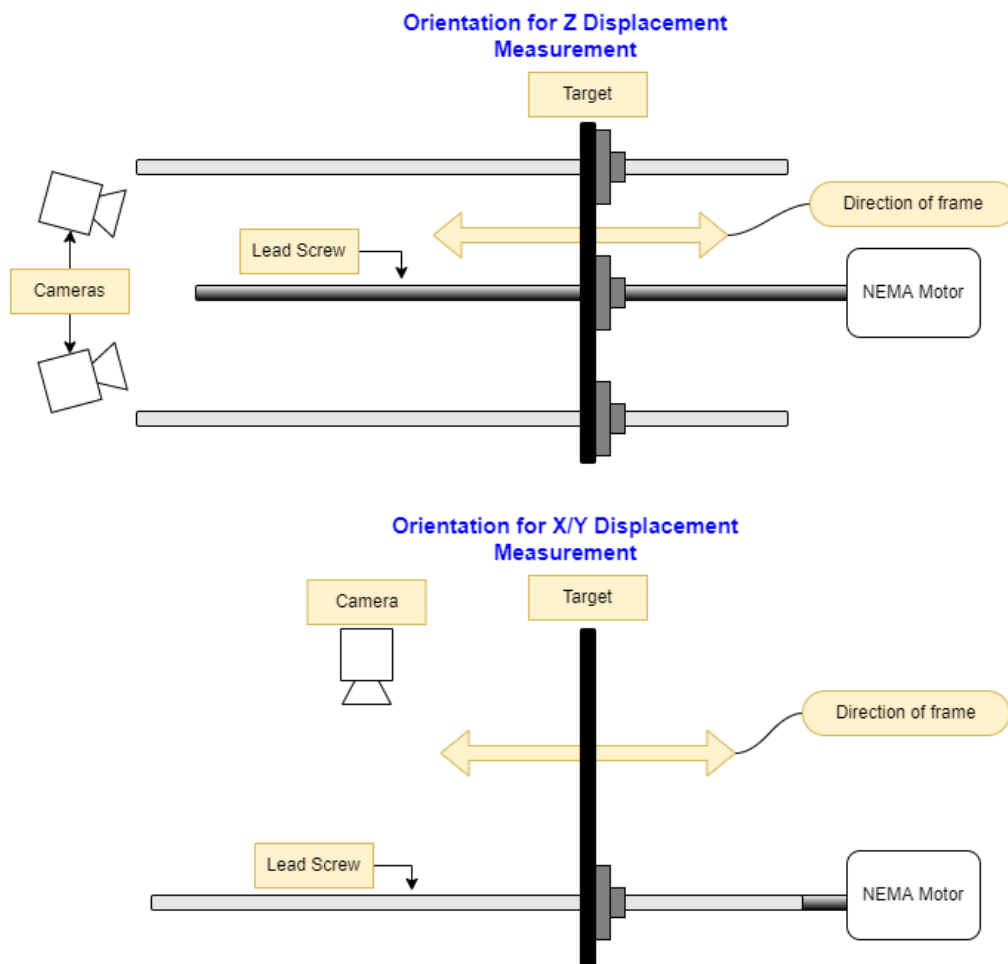


Figure 26: Camera Orientation for Different Axes Measurements.

3.3.2.1. Specimen Displacement Hardware

Since this was not part of the original scope of the project, the simplest and most practical approach to develop the experimental jig in question, was chosen. Further experimental setups would be based on the same basic system.

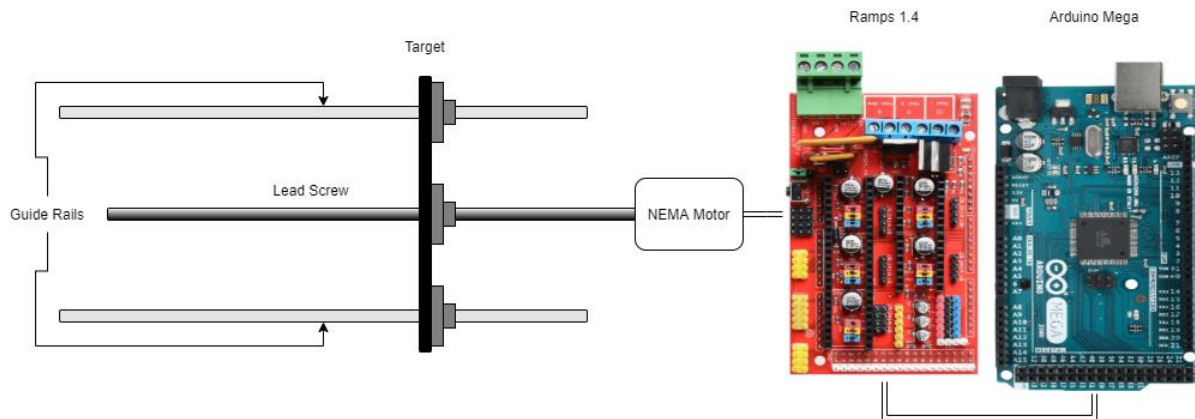


Figure 27: Experimental Design Setup.

The **RepRap Arduino Mega Pololu Shield**, or **RAMPS**, was created for the 3D printing space and allows for an interface between the Arduino Mega and the hardware required for 3D printing. The generally accepted motor for this application is a NEMA stepper motor (although others can be used). This collection of hardware substantially exceeded the requirements for the test regimen but was cost effective enough to justify its use. The Arduino language also provided a very simple and effective way of creating the *sketch* required to satisfy the requirements of the test jig.

The basic requirements for the test jig were to have a resolution good enough to run the experimental verifications required and to allow for control over a serial interface. Since off the shelf components were used the resolution cannot be prescribed and would have to be confirmed by experiment. This was so that the developed DIC software platform could direct the motion of the test jig. This would allow for an experiment to be run in a repeatable fashion with slightly different experimental inputs.

A 300mm lead screw was attached to the target with a brass nut. The lead screw was then attached to the NEMA motor with a flexible coupler shaft. Guide rails were created with 8mm round bar and linear bearings attached to the frame. To mount the guide rails to the platform mounting brackets were modelled and 3D printed.

A frame was then created to hold a sample that would be deformed. This frame has an area to clamp a specimen for deformation.

The ultimate objective of the experimental phase was to test whether this approach could be applied to a bulge test experiment. Due to the limitations of the available test infrastructure as well as the determinism this would provide, it was decided to use an indenter as opposed to an actual bulge test. The indenter has a known radius produced by CNC machining to a fine

tolerance. Hence the Z-displacement profile of the material in contact with the indenter would be known to a high accuracy without the need for an independent measurement.

A piece of flexible material (i.e., latex) would be placed into the clamp and an indenter would be moved to deform the material. To ease the development, it was decided to move the frame over the indenter and not the other way around. This makes little difference except for the direction of the Z displacement.

Figure 28 shows the final test jig with the indenter in place.

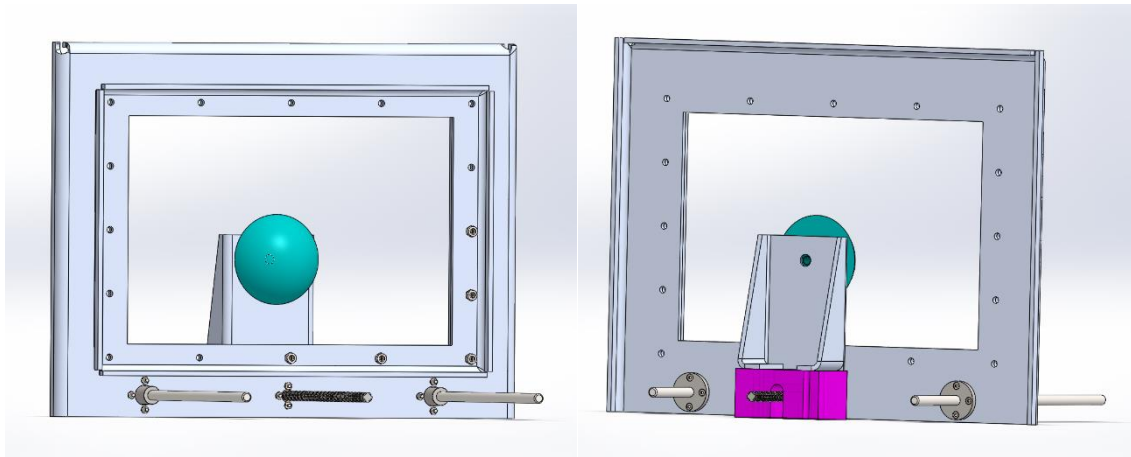


Figure 28: Test Jig with Indenter.

For the deformable surface a piece of latex was used. For ease of acquisition, a TheraBand® latex rubber band of thickness 0.15mm was used. This was speckled using spray paint.

The application of the speckle pattern was challenging due to a lack of appropriate tools. Ultimately standard enamel spray paint was used for the speckle pattern and applied directly from the spray paint can. Techniques for applying an even speckle from a spray can, with small enough features to differentiate subsets was experimented with and although certain techniques worked this proved not adequate. It was decided to use multiple colours to further allow the feature algorithms to pick up differentiations.

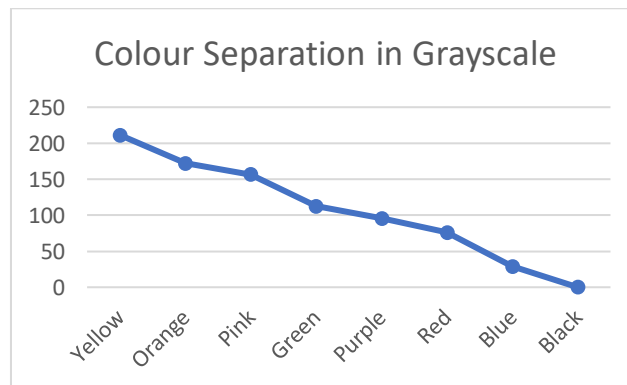


Figure 29: Grayscale Colour Separation

The algorithms for DIC use grayscale data to run the feature extraction. Values were chosen that were significantly separate in grayscale to allow for clear distinction in the algorithm. The base colour of the Theraband® was yellow. As a result, black was chosen as the best colour for the first layer. Red and green were chosen as further options that would then give second and third layer distinction to the grayscale algorithm.

The final hardware setup is shown in Figure 30.

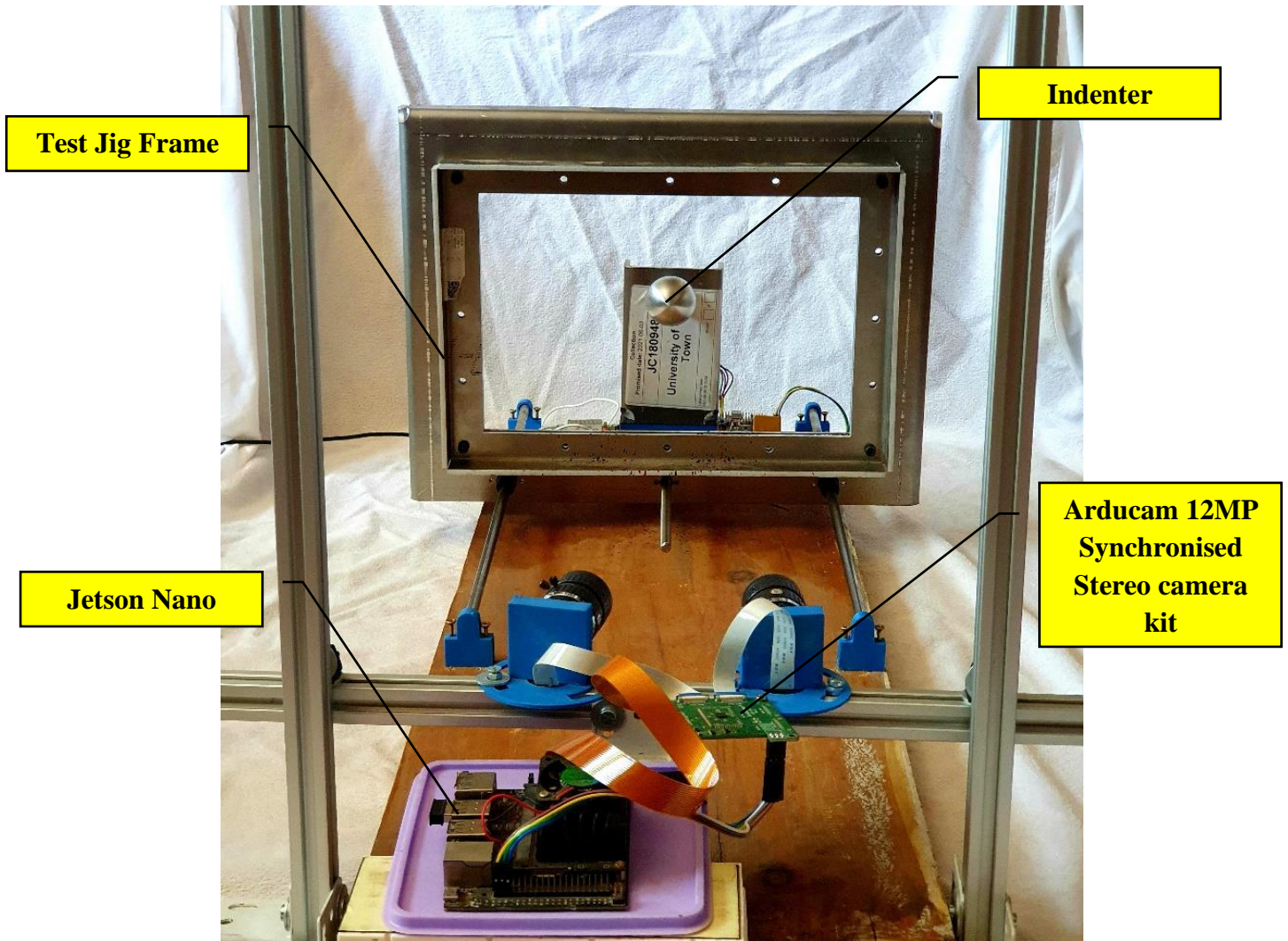


Figure 30: Final Experimental Hardware Setup.

3.3.2.2. Frame Movement Software

The flow diagram for the *sketch* running on the Arduino is given below.

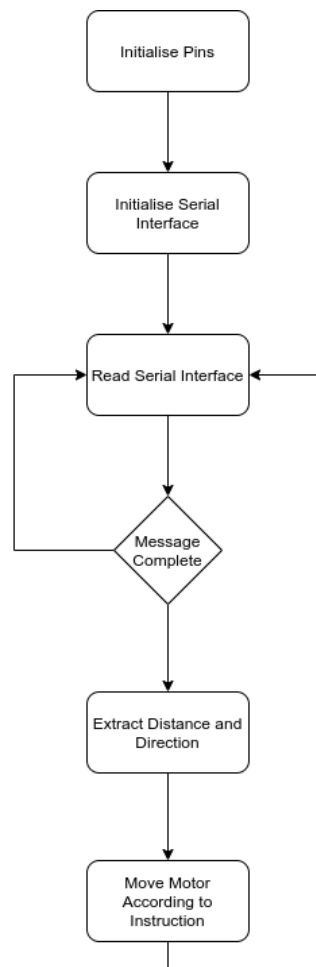


Figure 31: Test Jig Program Flow.

The Arduino waits for a serial message telling it to move. The message tells the Arduino to move the motor forwards or backwards at a specific speed or a specific distance.

From the main user interface, the plus and minus keyboard strokes will send the serial message to the Arduino to move forwards and backwards 1mm.

In order to make the experiments repeatable and require little intervention from the user it became clear the ability to control the test platform from a script was important. This would give flexibility to conduct various types of tests without the need to recompile the code each time. Various options were briefly considered such as [Duktape](#) (a javascript engine) or Forth (a stack-based language that is fairly easy to implement). However, given the limited functionality required, it was decided the scripting did not need to be Turing complete and a basic command-based script structure was decided on. These scripts are loaded at start-up and can be run after the DIC algorithms are initiated.

For convenience it was decided that the software needed to plot graphs of the results. Various options were again considered for this:

- [Sciplot](#)
- [ROOT](#)
- [Gnuplot](#).

Sciplot was decided upon because of its very simple code base and ability to completely integrate at a source code level into the project.

Note: ROOT is a piece of software developed at CERN, that in other applications is worth investigation. Its complexity and feature list made it inappropriate for this project, but the code base is well done and feature rich and is worthy of a mention.

3.4. Design Description

The final architectural structure of the software is defined in Figure 32.

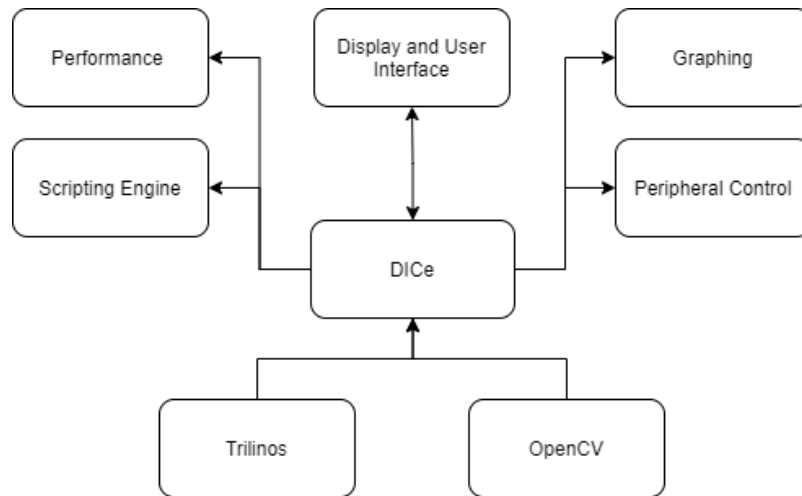


Figure 32: Final Software Architecture.

The core of the system would need to be the DIC algorithm which would be configured and run within the DICE software framework. The support elements required to enable the experiment are:

- *An image display and capturing system:* OpenCV was chosen for this – this package is already a dependency of DICE so using other functions within it was a trivial decision.
- *A system for control of peripherals:* The experimental setup will need to control stepper motors while running the experiments.
- *A scripting engine:* The system will need a way to automated tests that will be of a repetitive nature.
- *A means of performance measurement:* Since this project involves, in part, figuring out how efficient, timewise, these algorithms are, a means of accurately measuring the timing within the code is required.
- *Graph feedback:* Although not crucial to the overall output, a graphical representation of the data is still important. This will feedback the results of the test in near real time.

Figure 33 is a flow chart of the main execution paths of the code base, as developed. There are four main modes that the system can executed.

- Subset Creation (*s*)
- Main DIC algorithm (*i*)
- Calibration (*c*)
- Run Script (*r*)

It is worth noting that most 3D DIC systems will sample the images on a given sync clock, hence the sample rate is driven by the sync clock. In our case since the cameras are always sampled synchronously (by design) there is no external driver of the sample rate of the algorithm. As soon as the system can sample and analyse the next image it does.

The final code flow is as follows:

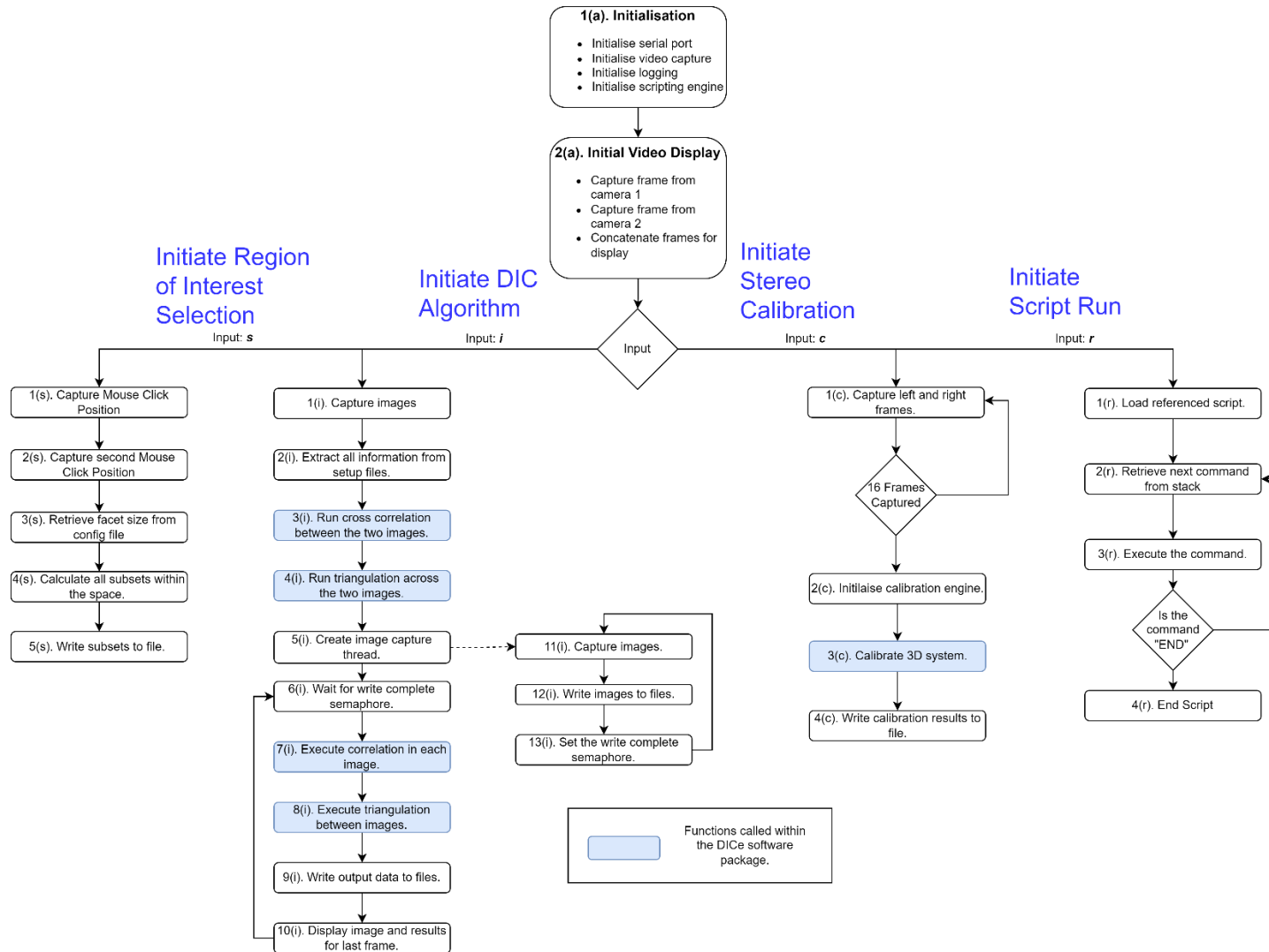


Figure 33: Final Code Flow.

3.4.1. DICE Interface Logic

DICE is a very modular piece of code. Out of the box it can be used as is since it has a GUI that one can use, written in Electron. Under the hood though each functional block can be broken down to a function call that can be executed from anywhere. This was the feature used in this project. The section below is not an exhaustive list of all functionality used in the DICE code base. Instead, it focuses on the large (and important) functional blocks that were used in the package. It is important to note that each of these functions would require extensive data preparation and setting up in code before being used. To get the details of this please refer to the code base associated with this project. What follows is a brief description of the important steps in Figure 33.

3.4.1.1. Step 2(i)

The software allows for the use of 4 main files as input parameters for the DIC process:

- A main input file detailing subset size, image locations, output folders and well as the location of other information.
- A parameters file (location defined by the main input file) that describes thresholds and settings.
- A calibration file (location defined by the main input file) that defines the calibration data for stereo camera systems.
- A file defining the subsets being evaluated in the DIC process.

All these files were used as is since there was no need to update their structure. The DICE libraries are then used to build the image sets, get their dimensions, load the calibration parameters, and set the output parameters.

3.4.1.2. Step 3(i)

The DICE libraries "execute_cross_correlation" and "save_cross_correlation_fields" are called to cross correlate subsets across the two images, using feature extraction algorithms, and ultimately save this field data.

3.4.1.3. Step 4(i)

Once these similar areas are found in both images the stereo set is triangulated to create the 3D model of the space.

3.4.1.4. Step 7(i) and 8(i)

After the initial correlation and triangulation is done to establish the 3D frame the system runs a 2D image correlation and triangulation to find the motion in each subsequent frame.

3.4.1.5. Step 3(c)

The calibrate function is called to try and calculate the parameters associated with the camera positions in space. The calibration logic is described in more detail in section 3.4.4.

Various other pieces of code that will not be mentioned here are used to write out results files, move large datasets around the code and extract required data from other datasets.

The full source code is available on GitHub. Please see Appendix 4 for details.

3.4.2. Image Capture

Obviously, the basis of any DIC algorithm is the images of the target that are captured. The image capture process changed in the middle of this project as a piece of hardware became available that was a much better fit to the application.

During initial development two 8MP USB camera modules were used to capture these images. Since DICE is based on OpenCV the capturing of images from webcams is trivial. On a Linux system, webcams come up as Video4Linux (v4l) devices and OpenCV can capture images from v4l devices with one function call. The image quality from webcams varies greatly and so for this initial development phase two matched webcams with external lenses were chosen to allow for greater control of the image quality. Fundamentally (especially for 3D stereo work) webcams will always suffer from the problem that in a standard hardware system (such as the one we are using) the webcams cannot be sampled at the same time; they would have to be sampled one after the other. This means that there is no guarantee that the 3D landscape the first sampled camera is looking at is exactly the same as the second. For this initial phase great care was taken to make sure that these cameras were sampled immediately after each other but, however negligible the concern, this is an inherently error prone system and will limit the deformation speed that can be accurately captured.

In section 3.1 it was mentioned that during the development of this project a piece of hardware was found that would allow for synchronised images to be captured as well as being hardware that could make use of the MIPI CSI interface on the Jetson.



Figure 34: Arducam 12MP Synchronized Stereo Camera.

The USB webcams were replaced with the Arducam 12MP synchronised stereo camera system shown in Figure 34. This was a cost-effective system (approximately \$150) and its availability was good, but its interface was non-trivial. Unlike a webcam this device does not come up as a v4l device which means that integration into the DICE OpenCV data stream required some specific interface development. For integration the gstreamer open-source framework was used [80]. Gstreamer allows for arbitrary media pipeline development. As shown in Figure 35 the image data is sourced from somewhere. It can then be altered in format, size, orientation etc and then the output sunk to a variety of sources.

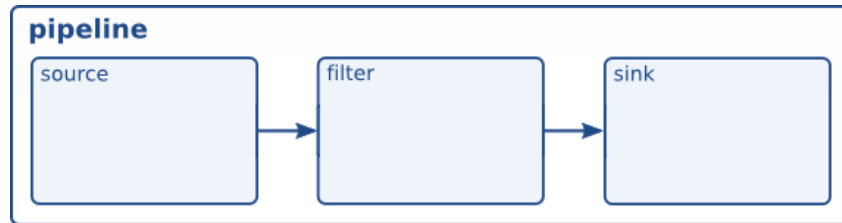


Figure 35: Gstreamer pipeline development [81]

The full gstreamer pipeline is noted for reference:

```
nvgarguscameraSrc ! video/x-raw(memory:NVMM), width=(int)4032, height=(int)3040, format=(string)NV12,
framerate=(fraction)13/1 ! nvvidconv flip-method=2 ! video/x-raw, width=(int)2048, height=(int)1536,
format=(string)BGRx ! videoconvert ! video/x-raw, format=(string)BGR ! appsink
```

An extra step was needed when using the Arducam in that to sample two cameras simultaneously the hardware for the camera splices the two images together and provides them to the CSI bus as one camera image. As a result, once the image is in the DICE OpenCV data stream the images need to be split apart into the left and right camera views.

An input parameter to the software was created so that the system can use either the webcam or Arducam interfaces transparently.

A processing thread was created to read images out of the cameras and write the files to the system for use by DICE. This thread attempted to optimise the flow of the code by using semaphores to signal the completion of the process. This is encapsulated in steps 11(i) to 13(i) in Figure 33.

A final step in the process involves taking the output of the cameras and converting it to grayscale. The output of the cameras is standard RGB which needs to be converted as an input to the DICE algorithms. To do this the OpenCV colour conversion algorithms were used. These perform a standard colour conversion as shown below:

$$Gray = 0.299 * Red + 0.587 * Green + 0.114 * Blue$$

Equation 13: Grayscale Conversion Equation

3.4.3. User Interface

The first objective of this project was always to investigate the implications of producing a system that could do near real-time DIC. To leverage any benefit from a real time system the interface would be required to show some representation of the information being gathered in real time. Two approaches were decided on:

- The system would display a heat map of the subsets being investigated, where the heat map would represent displacement (in a chosen axis) with varying colours (from blue to red).
- The system would output the displacement, in all axes of a sample, of the subsets to show the progress of the measurements.

The final system interface is shown in Figure 36.

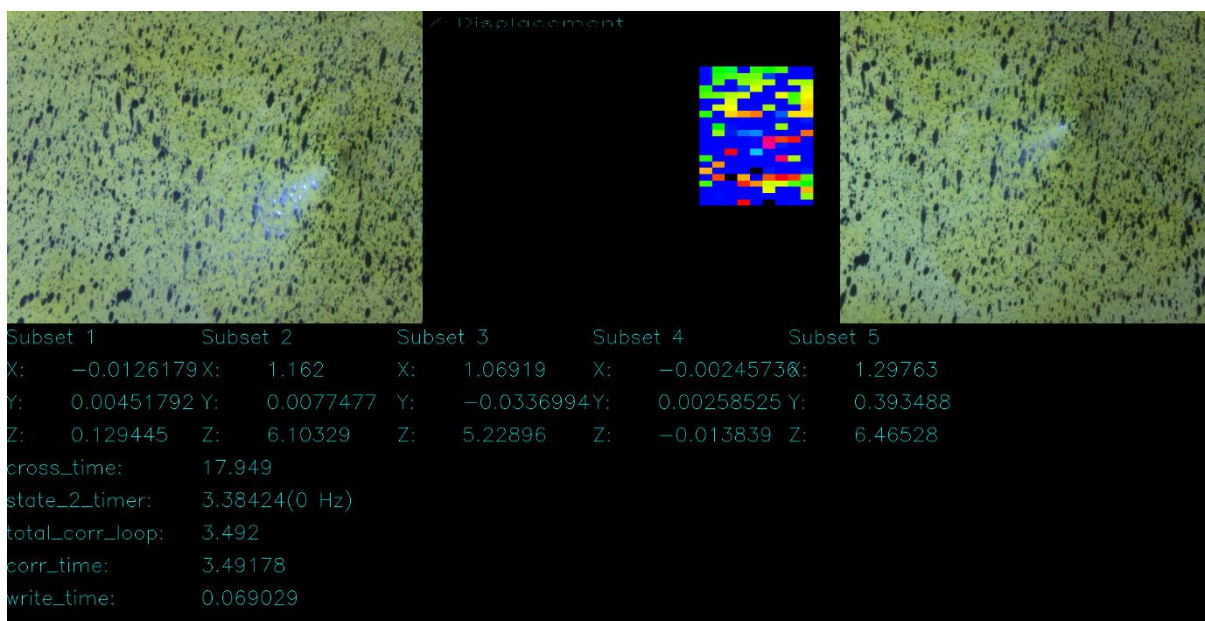


Figure 36: Final User Interface.

The left and right images from the camera are shown in the top left and right respectively. This particular image was taken while an indenter was pushing on the latex surface. Below the images a sample of 5 subsets' data is shown for all axes. Some other performance metrics are included under the subset data. This information encapsulates the speed of the algorithms.

Between the two images is a heat map of the displacement. Red corresponds to larger displacements, with blue being smaller displacement. Initially the idea was to display the subsets overlaid on the actual image. The principle was that in each subset (when displaying the image) the code would split the image data in the subsets into the red, green and blue components. The values of these three channels, on each pixel, would then be manipulated to overlay a representation of the displacement while retaining the image data underneath. The idea was sound, and the implementation did work, but it was found that the processing involved in splitting the image into channels, manipulating the data and then re-merging the channels to form an image was too time consuming to achieve any type of real time response. As a result,

a separate heat map was inserted between the two images. Since this requires only the drawing of the subsets with the correct colour for the displacement it proved much more palatable from the processing power perspective and did not significantly impact on the processing time per frame.

The user interface is keyboard-press based. The following keyboard commands initiate the actions described in the application.

Keyboard Press	Action
c	Start the calibration process.
q	Quit the application.
i	Initiate the DIC process.
r	Run the script loaded at startup. <i>The script is a file passed to the program at startup as a parameter. It must be defined.</i>
s	Start the subset selection process <i>This process requires the use of the mouse as well as it captures a rectangular area for the area under investigation. The rectangular area is marked by the mouse click and mouse release.</i>
x, y, z	Tells the system which axis to show in the heat map and which axis to graph at the end of the test.
m	Create a movie – in captures stills – of the heat map as the test proceeds.
+	Run the target forward 1mm.
-	Run the target back 1mm.

Table 4: Keyboard Command Table.

3.4.4. Calibration Logic

There are many ways to calibrate stereo cameras. The easiest way (and the way that DICe has implemented it) is to take pictures of a known target in various poses and then to calculate the extrinsic and intrinsic values of the camera system as discussed in section 2.4.1 and 2.4.2.

DICe allows for calibration with a dot pattern or a checkerboard pattern. Both were tried and ultimately the checkerboard pattern was chosen purely because it is easier to actually make the calibration target.

To setup DICe for this an input file is passed to the calibration algorithm. In Figure 33 this is step 2(c). This file contains:

- The folder that contains the calibration images.
- The image range (how many images and how are they named).
- The setup of the checkerboard (number of squares and their size).
- Settings of the calibration (such as calculating extrinsic and intrinsic parameters and, fixing the principal point).

Once the setup information has been passed to the algorithm the calibration images need to be captured. Initially the software captured 16 images in sequence with no feedback. This worked but resulted in a lot of wasted time. If the images were not good enough to resolve the corners

(in the case of the checkerboard) in every image the quality of the ultimate calibration was questionable.

The final implementation of the calibration process tries to resolve the target corners on every image capture and displays a green circle indicator if successful. This lets the user know that the image captured is suitable for calibration purposes. This process takes a bit longer since parts of the algorithm need to run on every capture, but it ultimately gives the user a better perspective on whether the calibration will be successful and accurate. In Figure 37 the calibration image is shown during the process. Note that the corners have been marked as found by the green circle. This process gives the user confidence that the image is useful for the final calibration algorithm.

The quality of the calibration is estimated using two metrics. The DICE engine uses the OpenCV stereo calibration algorithm to perform the calculation with two stereo cameras. This function returns the final value of the re-projection error. Ultimately this is an RMS value of all re-projection errors for a set of selected points.

Further the DICE algorithm calculates the quality of the calibration internally by using the epipolar geometry constraint to calculate an epipolar error for a set of points. The value of this error is output to a stored file as part of the calibration process. All experimental data included in section 4 includes a file “cal_error.txt” which contains the epipolar error for a set of points.

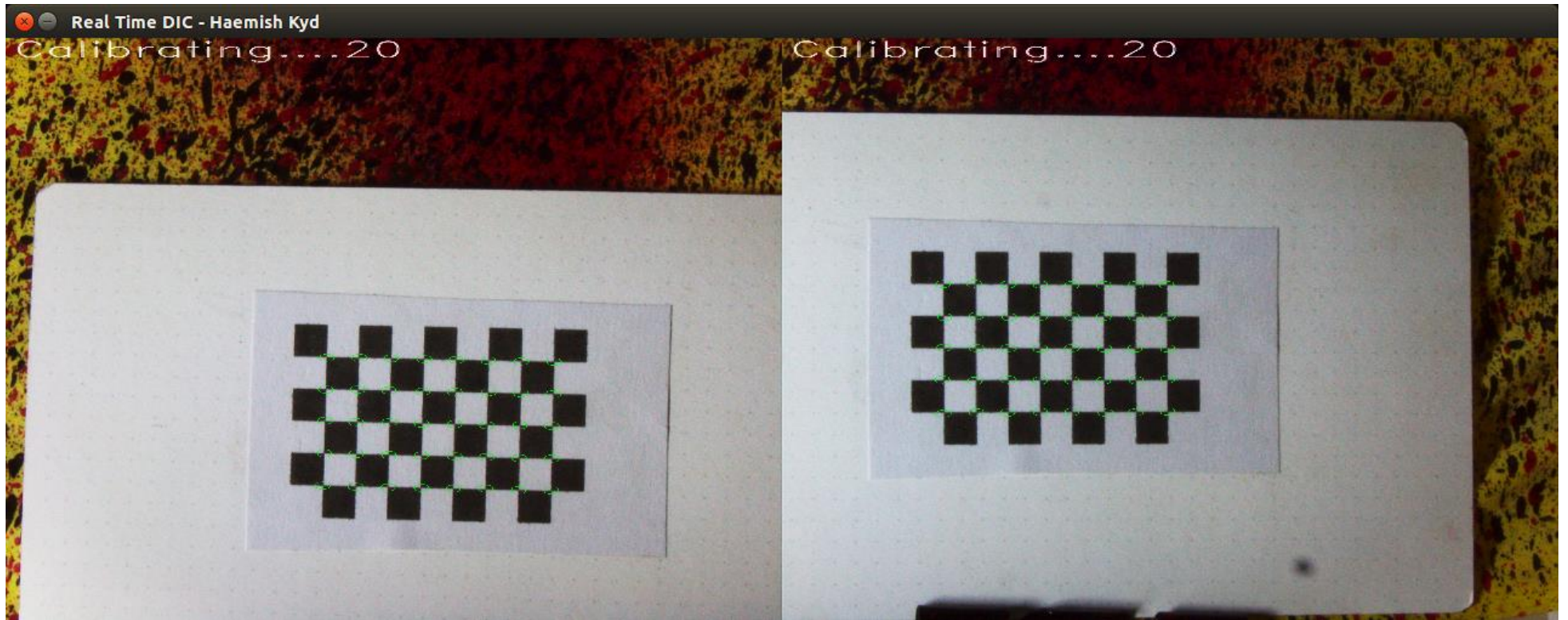


Figure 37: Calibration capture process (showing analysis of corners).

4. Experimental Data

With the system developed and test jig created the research questions proposed by this project may now be addressed. As a reminder, these questions are:

- Given a specified image resolution and frame rate, can a 3D DIC system yield acceptable accuracy of deformation measurements when running in near real-time?
- What implications arise from the constraint that widely available and cost-effective hardware and software be used?

To answer these questions, the experimental plan described in Table 5 was followed.

All of the experiments in this section have the mentioned data available for review online. The data is stored in a github repository and is available for download and review, at:

https://github.com/haemishkyd/kydhae001_msc_results

The data can be viewed online at the address above by clicking on the “*.ipynb” file in each folder.

It is possible to analyse this data on any computer – please see Appendix 4 for details on how to do this.

For these experiments the axes are as noted below:



Figure 38: Frame of Reference for Experiment.

In the diagrams above the direction of the arrow denotes the positive direction. This is true for all experiments except in Section 4.4.4 where the axes are deliberately inverted to illustrate the ability of the system to change its frame of reference.

The experimental roadmap is shown in Figure 39. First a basic characterisation of the noise of the system will be completed. The varying effects of subset size and number will then be considered. Having established these basic principles, the accuracy and speed of the system will be determined in controlled experiments.

Lastly, and finally answering the question posed by this project, a real-time experiment will be conducted characterising a bulge using the 3D DIC framework developed in this project.

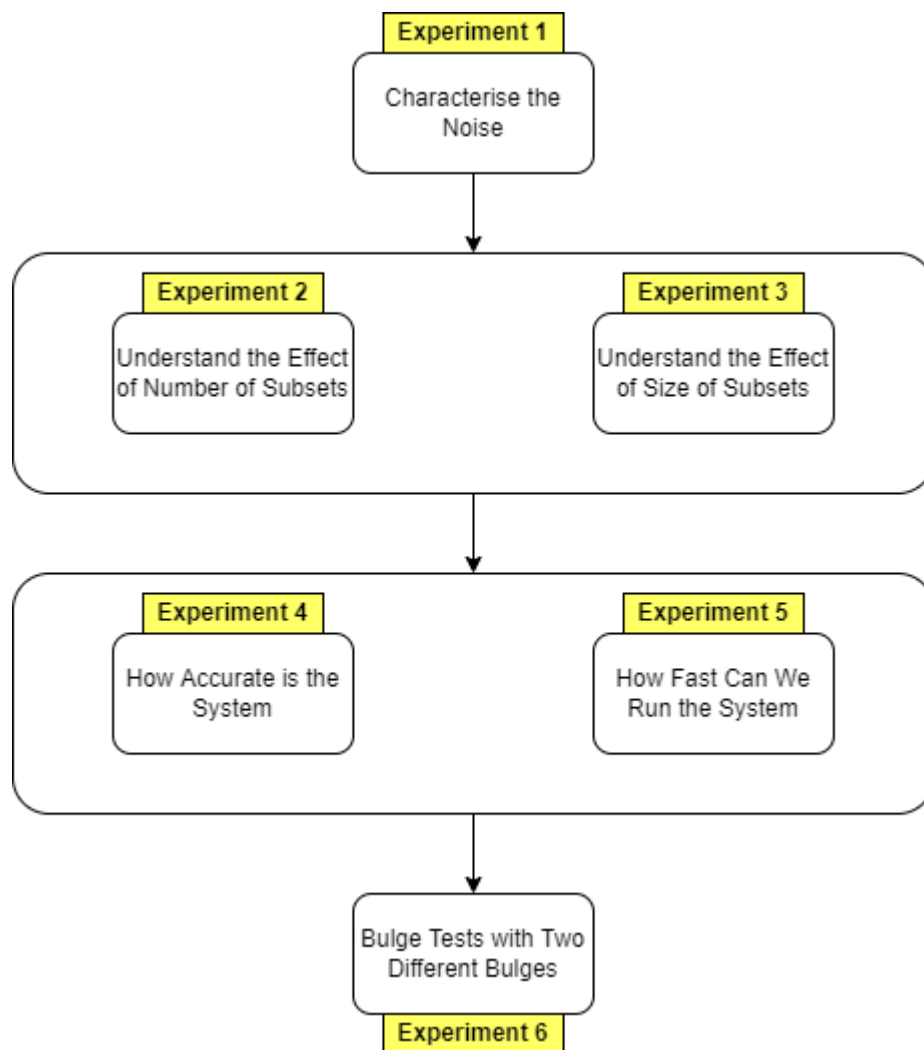


Figure 39: Experimental Roadmap.

Table 5 summarises the experiments executed to validate the research question. What follows is a description, basic procedure and results comparison for each of the experiments below.

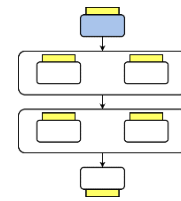
Experiment No.	Experiment Set	Description
1	Noise Floor Experiment	This experiment tries to understand the level of noise in the measurement of the displacement of the system. The outcome of this experiment tells us the highest level of accuracy we can (reliably) expect from the system.
2	Speed Experiment – Number of Subsets	This experiment attempts to determine the effect of adding subsets to the test. How does the addition of more area being correlated affect the speed of the process?
3	Speed Experiment – Subset Size	This experiment attempts to determine the effect of the size of the subsets on the speed of the process.
4	Motion (Accuracy)	This experiment tries to understand the level of accuracy we can expect from the system with a target with precisely known movements. Can the system keep track of the displacement and within what tolerance?
5	Motion (Speed)	This experiment attempts to understand the highest frame rate that we can expect when attempting DIC in near real time. How fast can we move the target while maintaining the accuracy of the measurement within the tolerance expected?
6	Indentation Tests	Spheres of known radius will be used to deform a flat, flexible surface to simulate a bulge test. This provides out of plane, varying displacements of a known profile for comparison with the DIC.

Table 5: Experiment Plan.

4.1. Experiment 1 – Noise Floor Experiment

4.1.1. Description

The goal of this experiment was to understand the noise characteristics of the DIC process for our experimental setup. Given the noise that is inevitable in the image capture and the fact that correlation is a statistical process we need to understand what the practical tolerance around this measurement is. For this experiment the DIC process was started but the target was not moved. In an ideal setup the displacement in all axes would be zero.



Data Set	Description
Noise/Noise_Floor_Sync_1	Run 1 of noise floor experiment
Noise/Noise_Floor_Sync_2	Run 2 of noise floor experiment

Table 6: Experiment 1 Data Sets.

4.1.2. Procedure

- Calibrate the stereo camera set.
- Create a subset input file with three subset regions.
- Run the DIC algorithm for no less than 60 seconds.
- Extract the MODEL_DISPLACEMENT_Z results for each subset from the DICe solutions file.
- Extract the MODEL_DISPLACEMENT_Y results for each subset from the DICe solutions file.
- Extract the MODEL_DISPLACEMENT_X results for each subset from the DICe solutions file.
- Clean the data for all data for each axis.
 - Remove any data points that are more than outside of $\mu \pm 2\sigma$
- Find the RMS of the data for each axis.
- Average the results across the data set for each axis. This is the noise floor for the given axis.

Various lighting options were tried in various poses during this experiment to highlight which type of illumination and at what distance it is most effective. This included:

- Incandescent bulbs.
- Fluorescent bulbs.
- LED bulbs.
- Circular LED lamps (ring light).

Ultimately the ring light gave the most consistent result and cast the most even lighting. The ring lights were placed on either side of the cameras and this distance was maintained for all experiments. The even casting of light is incredibly important since this experiment could eventually measure significant movement. If the light source is uneven, the target may move into less or more lighting which would compromise the measurement.

The source of illumination will be the LED ring light for all experiments going forward.

For this test three random subsets were chosen across the image. These subsets were monitored, and the displacement results logged. The subsets are shown in Figure 40, marked by the blue squares.

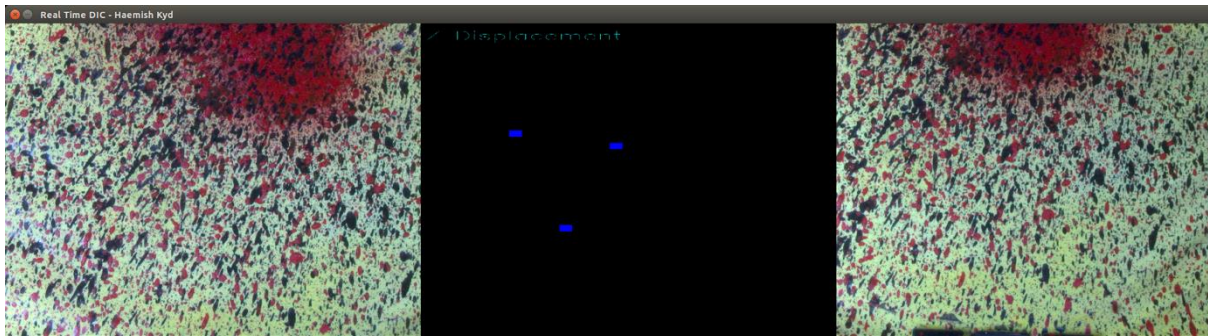


Figure 40: Illustration of subsets chosen for synchronised cameras (run 2).

4.1.3. Results

The motion of three subsets for each axis are given below. The experiment was run twice with different calibrations in different natural lighting conditions but the same artificial lighting.

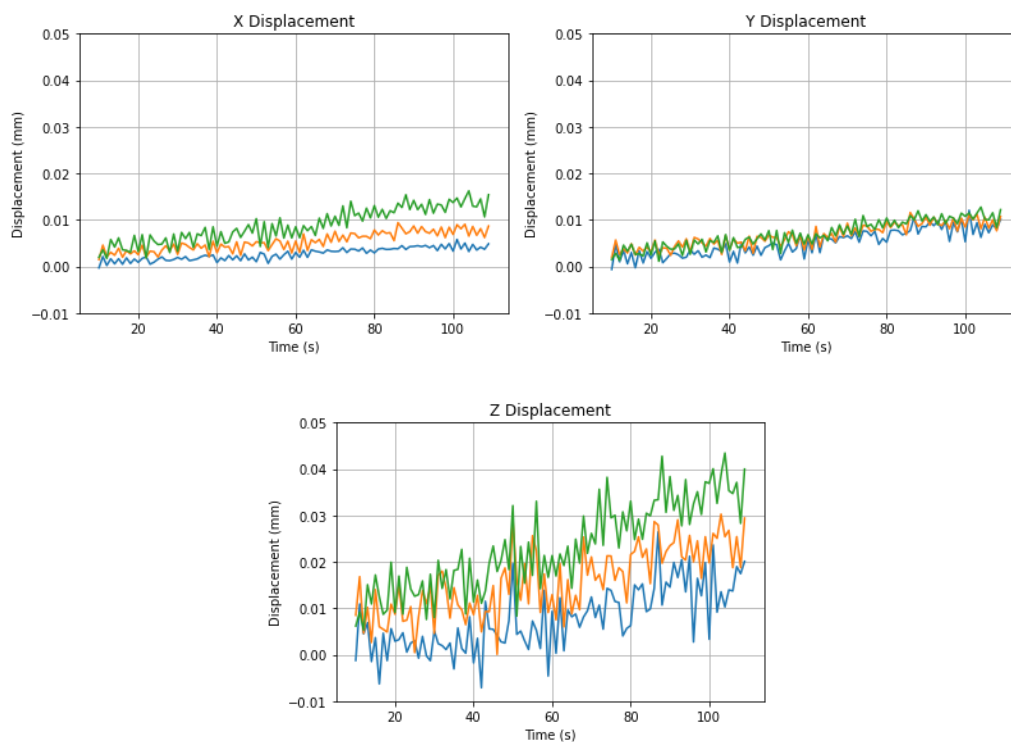


Figure 41: Graphs of motion of each subset in each axis (Run 1).

The results from Run 1 showed a trend that suggested slight motion in the system over time even though the system was stationary. Although the velocity of the trend was small enough to be negligible (a maximum of 0.01mm over 2 minutes) it was clearly there. There are various reasons that could be hypothesised for this behaviour.

- *Slight changes in environmental conditions.* If the temperature changes during the experiment the frame would expand/contract while the experiment was underway. These changes in temperature could also be found in the equipment itself, as the equipment is used the lenses and camera sensors would heat up. This would show as slight changes in X, Y and Z values that would track the increase or decrease in temperature. Further any changes in lighting would contribute to changes in displacement in unpredictable ways.
- *Correlation errors due to inherent structure in the speckle pattern.* If the speckle pattern is too regular or lacks randomness the correlation engine would start creating false positives for matches which would manifest as movement in the result. The algorithm that searches for matching patterns would follow an algorithm which could conceivably see the error always being realised in a certain direction.
- *Experimental Error.* It is conceivable that the target was indeed moving due to some or other very small motion in the experimental setup.

As can be seen in run 2 of this experiment, this behaviour was not seen again and as a result these hypotheses could not be tested. The results have however been included since they demonstrate that this *could* happen and that the process of establishing a noise floor can also serve the function of making sure that the system is stable before running an experiment that needs to be very accurate.

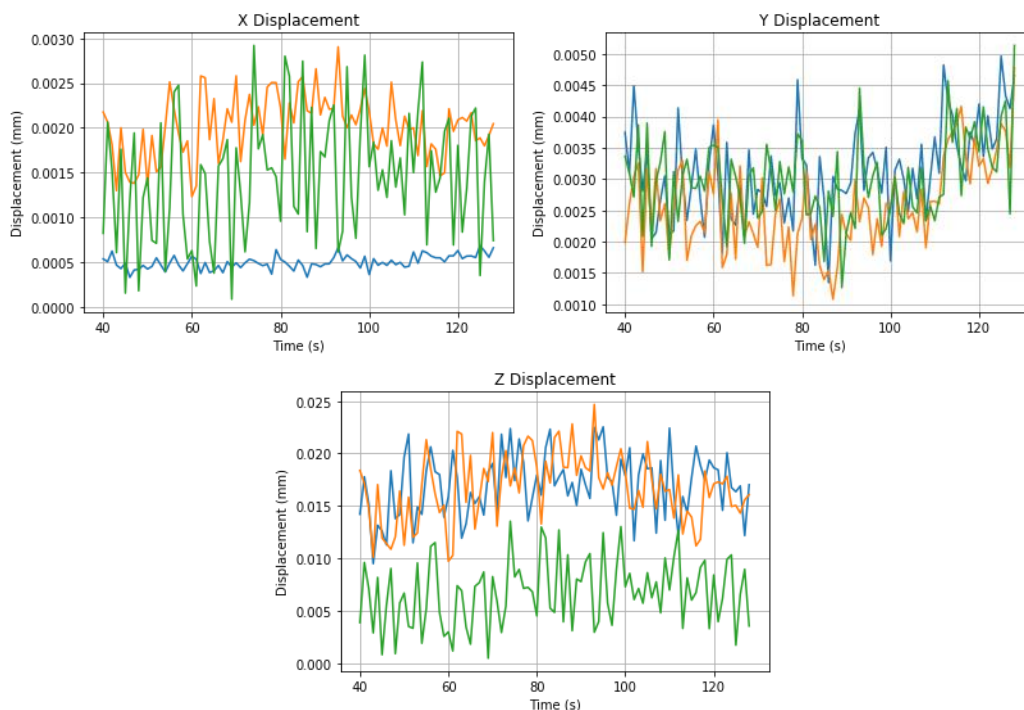


Figure 42: Graphs of motion of each subset in each axis (Run 2).

Run 1	Run 2
Subset 1 RMS : 0.00984 mm	Subset 1 RMS : 0.01734 mm
Subset 2 RMS : 0.01765 mm	Subset 2 RMS : 0.01702 mm
Subset 3 RMS : 0.02448 mm	Subset 3 RMS : 0.00736 mm
Avg Z Axis RMS: 0.01732 mm	Avg Z Axis RMS: 0.01391 mm
Subset 1 RMS : 0.00574 mm	Subset 1 RMS : 0.00316 mm
Subset 2 RMS : 0.00699 mm	Subset 2 RMS : 0.00262 mm
Subset 3 RMS : 0.00755 mm	Subset 3 RMS : 0.003 mm
Avg Y Axis RMS: 0.00676 mm	Avg Y Axis RMS: 0.00293 mm
Subset 1 RMS : 0.00295 mm	Subset 1 RMS : 0.0005 mm
Subset 2 RMS : 0.00567 mm	Subset 2 RMS : 0.00204 mm
Subset 3 RMS : 0.00946 mm	Subset 3 RMS : 0.00158 mm
Avg X Axis RMS: 0.00603 mm	Avg X Axis RMS: 0.00137 mm

Table 7: RMS Noise Results.

It is important to note that this result is a combination of all the factors currently in the system. This means that this result is only true for these lenses, in this light, with this length of cable between lenses and Jetson board.

Other than lighting conditions the rest of the components would *probably* produce similar noise results, but this cannot be guaranteed. The light level, if changed, would have a large effect on the noise level.

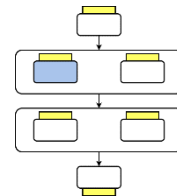
With the set of factors constant in this experimental setup, the results show that we can expect a measurement resolution for out-of-plane (Z) displacements of no better than ~0.015mm. The actual measurement resolution achieved may be worse given other factors in the experiment but from these results we can define an absolute best-case resolution of measurement. For the type of experiment (i.e., bulge tests) a measurement resolution of 0.015mm is more than adequate. If this was applied to any other type of DIC experiment the requirements of the experiment would need to be evaluated against this limit.

Very expensive DIC systems would have less variability across experimental setups. The cameras would be more consistent, with cabling made for purpose with predictable responses. Since this project aims to see if 3D DIC can be run using off-the-shelf components the variability across components would need to be checked. This noise floor experiment is the easiest way to determine the inherent inaccuracies in the system.

4.2. Experiment 2 – Speed: Number of Subsets

4.2.1. Description

To understand the extent to which we have achieved the goal of creating a platform to perform real-time DIC it is necessary to understand how fast we can actually perform the process. We would also need to know how changing the inputs to the system change this real-time performance. The speed of processing is closely related to how much image data must be processed, which is the product of the number of subsets and their size in pixels. In this experiment, the subset size was kept constant (at 30 x 30 pixels) and the number of subsets increased, to vary the amount of image data for processing.



This experiment tracks the speed of processing from 1 subset to 30 subsets and analyses the change in processing speed.

Data Set	Description
Timing/timing_1	Run with 1 subset
Timing/timing_2	Run with 2 subset
Timing/timing_3	Run with 4 subset
Timing/timing_4	Run with 6 subset
Timing/timing_5	Run with 8 subset
Timing/timing_6	Run with 10 subset
Timing/timing_7	Run with 12 subset
Timing/timing_8	Run with 14 subset
Timing/timing_9	Run with 16 subset
Timing/timing_10	Run with 18 subset
Timing/timing_11	Run with 20 subset
Timing/timing_12	Run with 22 subset
Timing/timing_13	Run with 24 subset
Timing/timing_14	Run with 26 subset
Timing/timing_15	Run with 28 subset
Timing/timing_16	Run with 30 subset

Table 8: Experiment 2 Data Sets.

4.2.2. Procedure

- Create a subset file with 1 subset defined. This subset can be randomly placed.
- Run the DIC algorithm with this subset.
- Store the timing.txt file created, for analysis.
- Create a subset with 2 subsets defined. These can be randomly placed.
- Run the DIC algorithm with this subset.
- Store the timing.txt file created, for analysis.
- Repeat with 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30 subsets and record all of the results.

4.2.3. Results

The average amount of time to process for the varying numbers of subsets is given below:

No. of Subsets	Processing Time(s)	Estimated Data
1	0.5408	1.7KiB
2	0.6626	3.4KiB
4	0.6811	6.8KiB
6	0.6446	10.2 KiB
8	0.6877	13.6 KiB
10	0.7208	17 KiB
12	0.7748	20.4 KiB
14	0.8056	23.8 KiB
16	0.9201	27.2 KiB
18	0.9315	30.6 KiB
20	0.9415	34 KiB
22	1.0662	37.4 KiB
24	1.0737	40.8 KiB
26	1.0770	44.2 KiB
28	1.1184	47.6 KiB
30	1.1700	51 KiB

Table 9: Number of Subsets vs Processing Time.

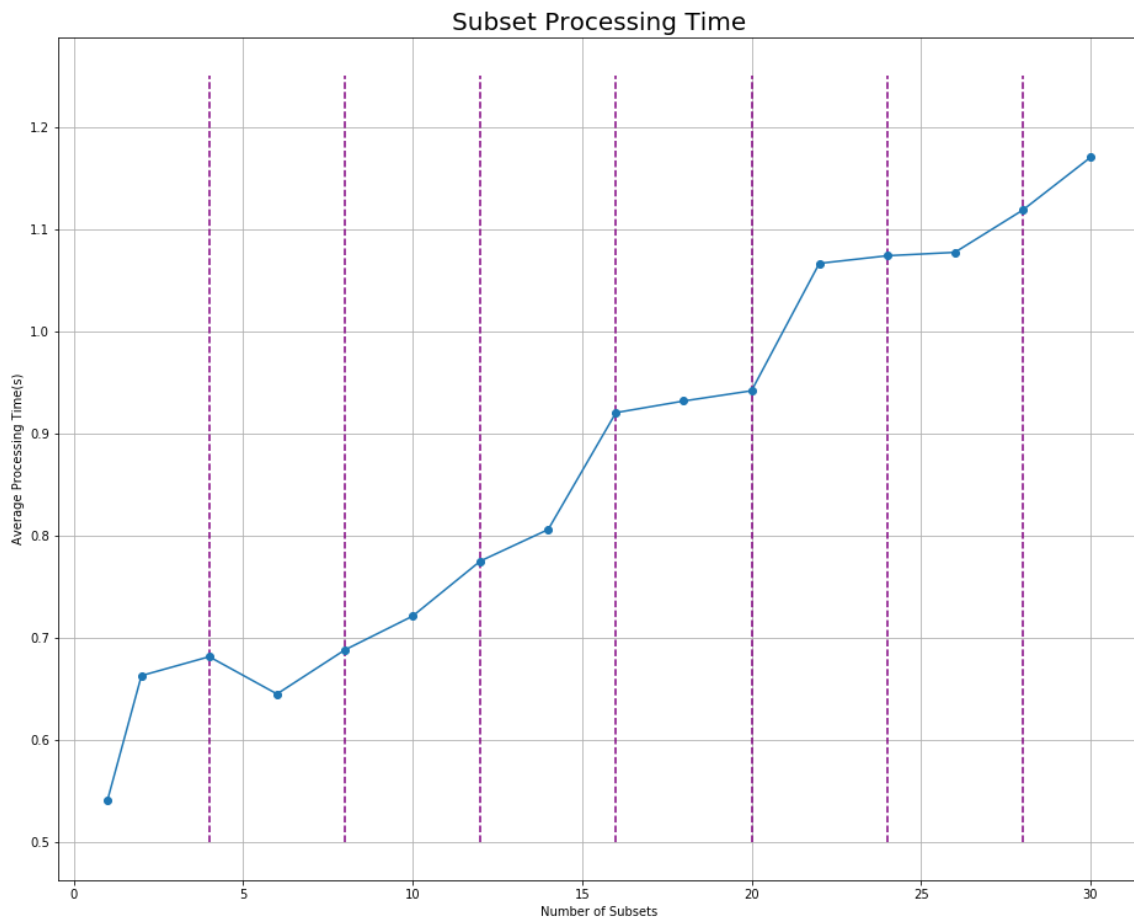


Figure 43: Processing Time for Varying Numbers of Subsets.

As would be expected, as the number of subsets increases the amount of time for the DIC algorithm to complete increases. In a single core system with a scheduler not subject to context switching, the graph would be fairly linear. In Figure 43 the trend is roughly linear, but has some noticeable deviations. There are two main reasons for this:

- The operating system that this software runs on is subject to context switching (as would any system that was not a low level RTOS)
- The processor here is a quad core device (*ARM® Cortex® -A57 MPCore (Quad-Core) Processor*).

The last point here is worth exploring a little further. The data does not support it outright, but there is evidence in the data (at 2 to 6 subsets, 16 to 20 subsets and 24 to 28 subsets) that the processing speed would go up in steps as the operating system passes the processing to the four cores available on this processor. Since this is a quad-core processor the graph highlights every four subsets (purple dashed line) to analyse if there is a correlation. There are places in the data where this stepped theory is not clear, but this could be due to various factors such as operating system interventions or peripheral interrupts. This is not the core question being asked here but it is an interesting point to consider going forward.

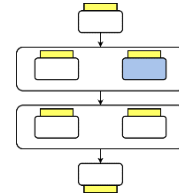
The parallelisation questioning could be taken a bit further. The DICe engine used in this work handles the processing of the subsets as part of its internal operation, i.e., how and when each subset is processed forms part of the underlying function of the engine. If the engine were stripped down a bit more, the individual processing blocks for each subset would be exposed. This would allow for a lot more exploration of the parallelisation concept and ultimately may lend itself to using the very powerful CUDA cores available on the Jetson Nano.

4.3. Experiment 3 – Speed: Subset Size

4.3.1. Description

In the experiment in section 4.2 the number of subsets was evaluated for its effect on the speed of execution. If the region of interest (ROI) is constant and the subset size is increased, would this then make a difference? In essence the question is, is it purely the number of subsets or the ultimate size of the ROI (and hence the total volume of pixel data) that dictates the speed of execution?

This experiment uses the same sized ROI while changing the size of the subsets within the ROI. The subset sizes chosen were 60, 30 15 and 7 pixels. In this experiment the subsets do not overlap so the larger the subset size the fewer subsets will exist in the ROI.



Data Set	Description
Subsets/subsets_1	Subset size of 60 pixels
Subsets/subsets_2	Subset size of 30 pixels
Subsets/subsets_3	Subset size of 15 pixels
Subsets/subsets_4	Subset size of 7 pixels

Table 10: Experiment 3 Data Sets.

4.3.2. Procedure

- A suitably speckled target was marked with start and stop points for ROI creation. These fixed corners for the ROI are shown with red circles in Figure 44.

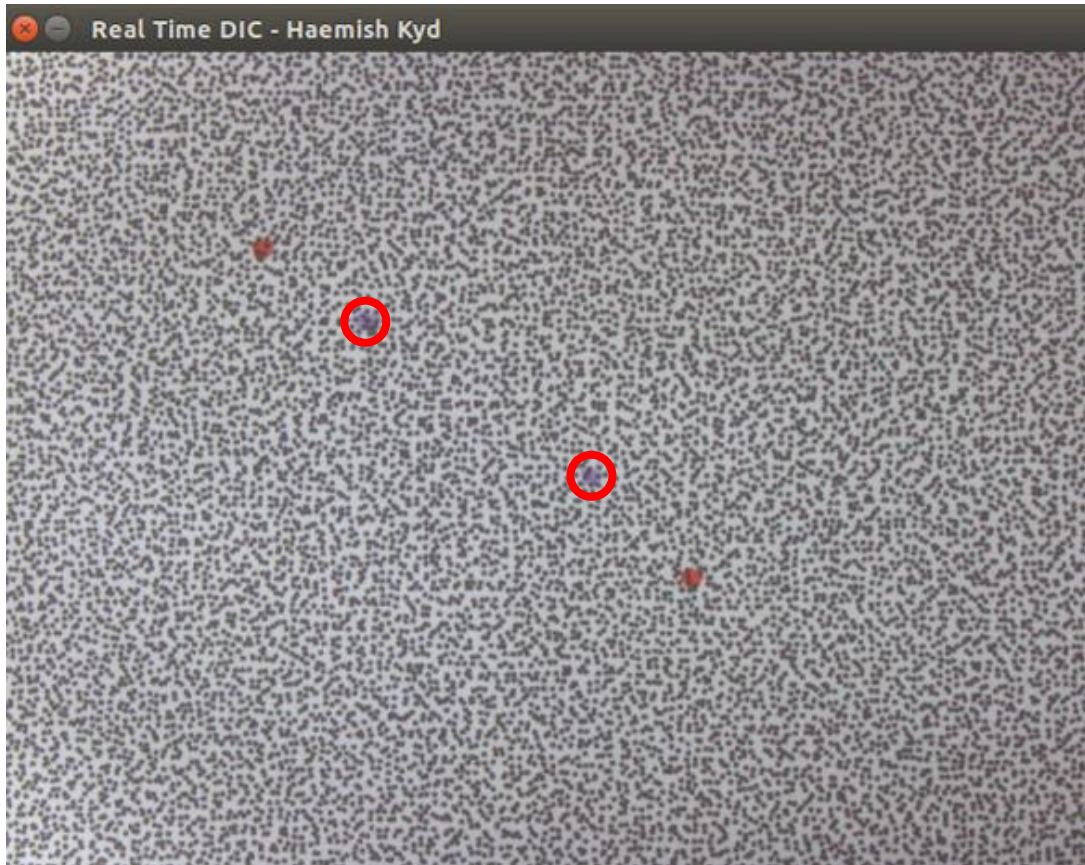


Figure 44: Region of Interest Selector - the blue dots were used.

- The input.xml file was updated to reference the new subset size.
- The program was started, and the software placed into “Region of Interest” mode. See Figure 33.
- The region of interest was marked for each subset size.
- The frame was advanced 10mm at 1 mm/s, then retracted to its original position at the same speed, to test the subset tracking.

4.3.3. Results

The average amount of time to process for the varying subset sizes is given below:

Subset Sizes (pixels)	Number of Subsets	Processing Time(s)
60	20	1.3853
30	70	1.3989
15	280	1.6417
7	1149	1.2187

Table 11: Subset Size versus Processing Time.

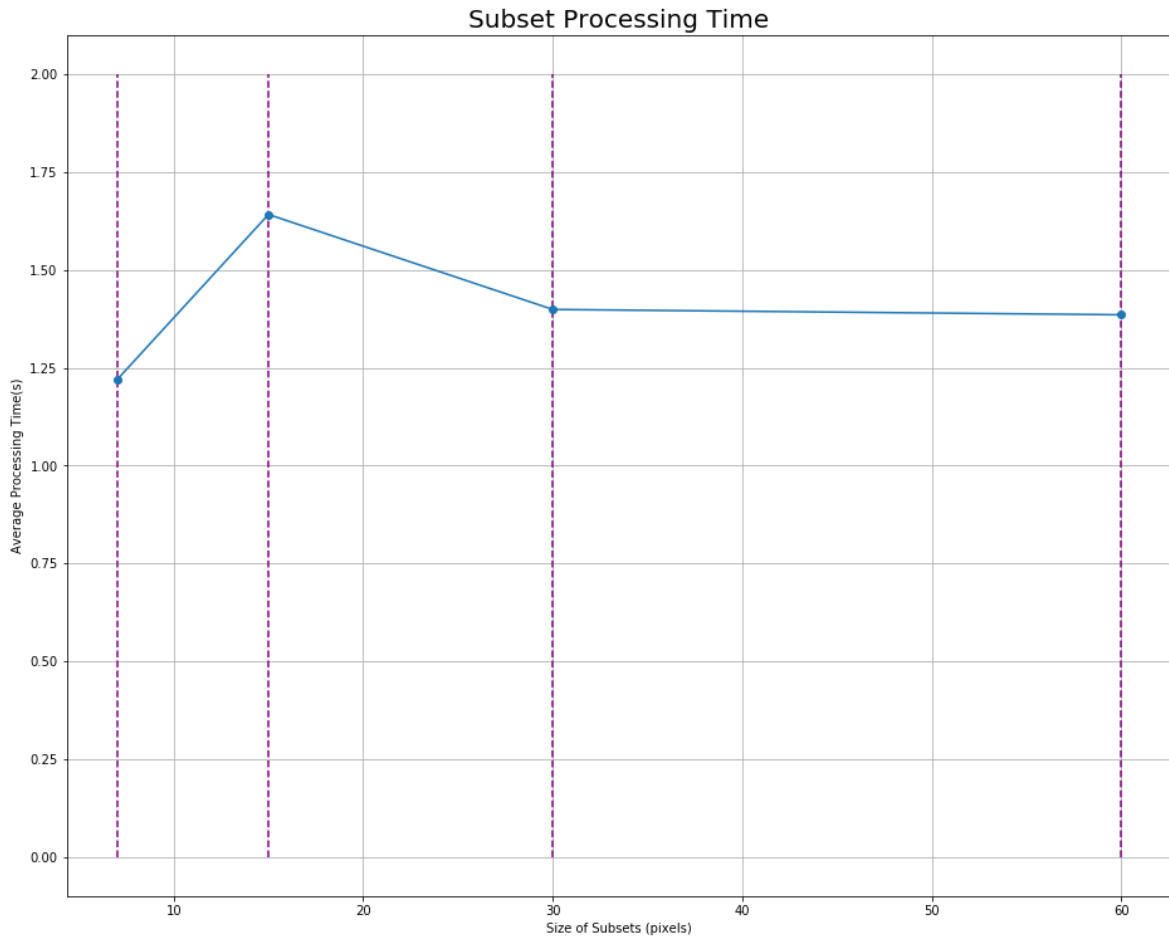
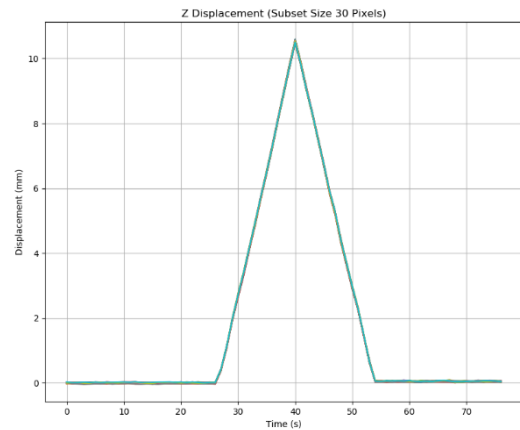
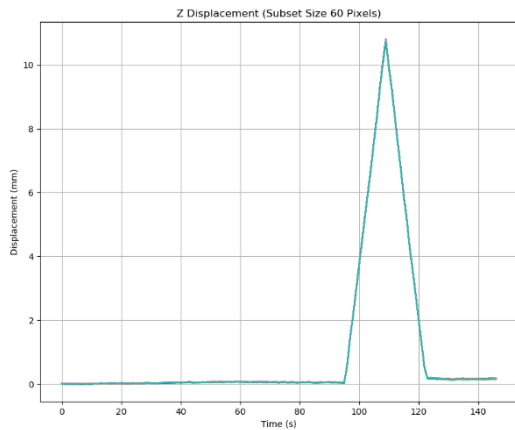


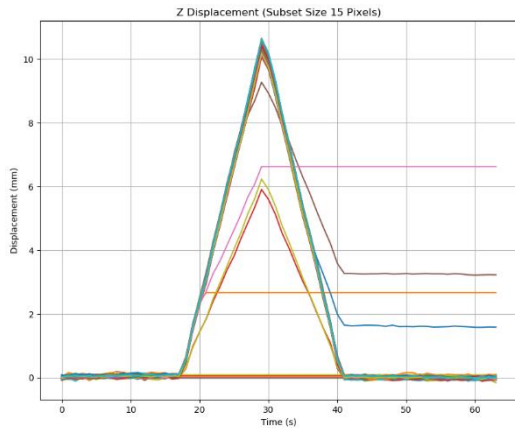
Figure 45: Processing Time for Varying Subset Sizes.

The results show if the size of the ROI is constant, the number (and by extension size) of the subsets for the given ROI has a negligible impact on the amount of time for the algorithm to execute.

The temptation would then be to keep reducing the number of subsets since the finer the granularity the more information can be gleaned about the deforming surface. To test this, the actual tracking of the subsets was graphed and is shown in Figure 46.

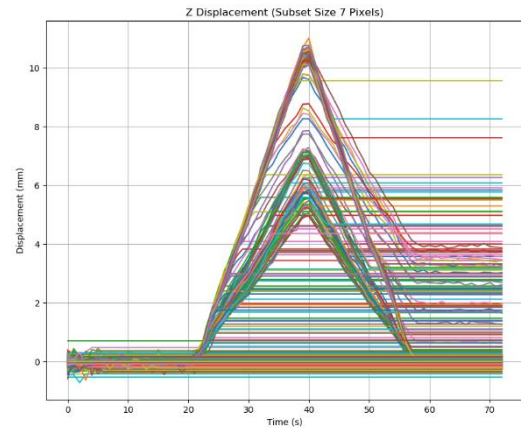


Subset size 60 pixels



Subset size 15 pixels

Subset size 30 pixels



Subset size 7 pixels

Figure 46: Z Displacement Response with Varying Subset Sizes.

This set of data shows that as the subsets get smaller the underlying speckle pattern stops being able to present a unique pattern to the algorithm in all instances. For a very small subset size the speckle pattern would need to be very dense and varied within that density. So it is fair to say that a smaller subset size comes with the complexity of implementing more and more intricate speckle patterns.

For the speckle pattern used in this experiment (refer to Figure 44) the subset sizes of 60 and 30 pixels worked well and allowed for tracking throughout the experiment. For subset sizes of 15 pixels, some subsets lost correlation during the experiment and for subsets of size 7 pixels many subsets cannot be tracked from the outset, with even more subsets losing tracking as the experiment progresses. The heat maps during the tracking showed this clearly, as can be seen in Figure 47. For these heat maps the zero displacement was blue so the subsets that remain blue show subsets that were not tracked from the outset.

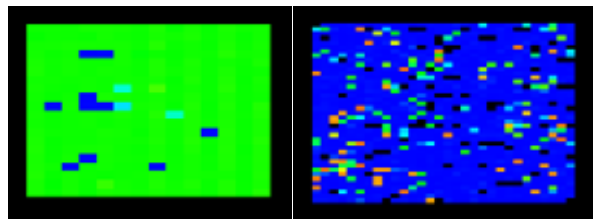


Figure 47: Heat map for subset sizes 15 and 7 pixels.

We have seen that subset size has a negligible effect on the processing speed of the algorithm. The same cannot be said of the total number of pixels. The amount of data being processed in each iteration of the algorithm is completely based on the pixel density of the camera. We are using 12MP cameras in this application which for DIC, given the distance to target and control of lighting normally associated with the experiments, is not necessary.

Had they been available would 5MP cameras (for instance) been better from a speed perspective? The simple answer is yes, we showed that the data processed is correlated to the

speed of processing in Experiment 2, but the degree to which this would be true has not been measured.

The region of interest is based on a physical object in the field of view, which means that for a given experiment a higher density of pixels would mean more data for the same region, i.e., *more* but *smaller* pixels. This means that for a lower density of pixels there would be less data to process (*fewer* and *bigger* pixels) which should result in faster processing speeds.

In Figure 46 we showed that the subset size had a substantial effect on the ability to distinguish the speckle pattern during the experiment. The smaller the subsets the less pixels and hence less distinguishing ability, the subset had. The larger pixels of the lower density camera would mean that for the same sized subset, the ability to distinguish the unique pattern presented by the speckle may be compromised. Consider the exaggerated example in Figure 48 below:

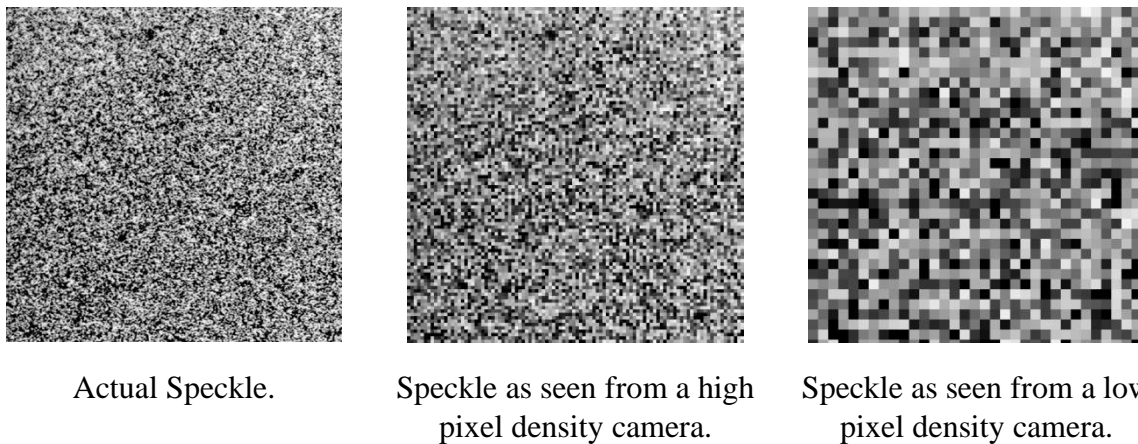


Figure 48: Pixel Density Comparison

Since the larger pixels would sample a larger area of the subset, the tracking of a pixel from frame to frame could be less accurate and hence compromise the correlation co-efficient achieved in the DIC process.

In short using a camera with a less dense pixel matrix would work to increase the speed of processing but there would be a point (depending on the target type) that the accuracy of the DIC tracking would start being compromised. This cost-benefit ratio would need to be gauged for each experiment.

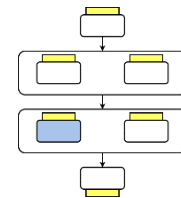
4.4. Experiment 4 – Motion (Accuracy)

4.4.1. Description

Whereas the real-time speed achieved is an interesting result, the actual accuracy of the measurement needs to be assessed. A system that cannot measure the actual X, Y or Z displacement reliably (i.e., within an acceptable tolerance) is not useful.

To this end great care was taken in the motion accuracy experiments to confirm accuracy of measurement with a few different test scenarios:

- Confirm the motion of the test jig.
- Confirm the measure of 2D displacement (in the Y axis).
- Confirm the measurement of a saw tooth profile with steps.
- Confirm the measurement of a slow continuous motion profile.



Data Set	Description
Motion/motion_1	Y displacement measured – continuous slow motion
Motion/motion_2	Saw tooth with frame of reference
Motion/motion_3	Saw tooth with frame of reference from the other side
Motion/motion_4	Z displacement measured – continuous slow motion

Table 12: Experiment 4 Data Sets.

4.4.2. Jig Motion Confirmation

4.4.2.1. Procedure

The accuracy of the measurement of the displacement will be determined with the test jig developed in Section 3.3.2.1. The test jig was built such that it would accept a command to move that would include the distance. The objective of this experiment was to make sure that when commanded to move a distance, the test jig accurately moves that distance within a tolerance.

A script was created to run the frame forwards 10mm in 1mm steps (given in Appendix 3). This test was run 5 times from an arbitrary starting position. Thereafter a new start position was chosen, and the test was run another 5 times. This experiment also tested the repeatability of the test jig. After the script moved the frame forward 10mm the frame was manually moved back 10mm. It should end up at the position it started.

The distance travelled by the frame was measured using a Vernier calliper and a fixed reference point on the base, for both the start and end positions.

4.4.2.2. Results

The results of the tests are given in the Table 13 below (all measurements in millimetres).

Start Position	End Position	Distance Moved	Start Position Error
195.41	185.85	9.56	-
195.48	185.8	9.68	0.07
195.41	185.72	9.69	0
195.42	185.69	9.73	0.01
195.33	185.66	9.67	0.08
Start Position	End Position	Distance Moved	Start Position Error
191.49	181.94	9.55	-
191.52	181.83	9.69	0.03
191.5	181.82	9.68	0.01
191.52	181.91	9.61	0.03
191.48	181.9	9.58	0.01

Table 13: Jig Motion Confirmation Results

The average distance travelled with a 10mm command is 9.644mm. The test jig can therefore be assumed to be calibrated to within 0.366 of a millimetre.

A point to consider in this result is that this test involved a stop/start motion: the frame would move 1mm, stop and then move 1mm again. This stop/start could cause small errors to accumulate, as on each step the frame had to overcome static friction in the linear bearings.

The average error on returning to the start position is 0.03mm. Refining the accuracy of the jig movement would be useful in this context but as ultimately 3D deformation would be measured with the indenter tests, which were independent of jig movement accuracy, this was not necessary.

4.4.3. Two-Dimensional Displacement

4.4.3.1. Procedure

The test jig was configured with the cameras pointing downwards. The target was a static speckled image and was set up such that the frame pushed the target forwards and backwards. A script was run that moved the target 10mm at a slow speed. This motion was in the Y direction in the frame of reference used.

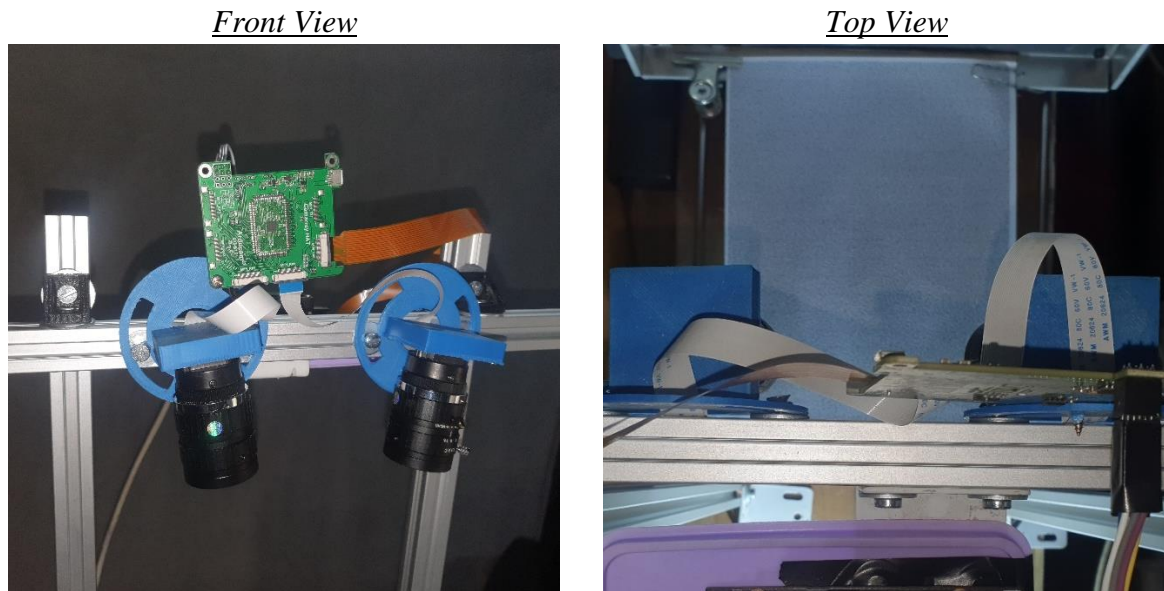


Figure 49: Y Displacement Measurement Equipment Setup.

For this experiment the displacement of 96 subsets were monitored. These subsets were distributed in a rectangle.

4.4.3.2. Results

The displacement as measured in the Y axis is displayed in Figure 50 below. The displacement measured was linear from 0 to 10mm. This is expected since the motion was constant. There are some small deviations to this linearity where the speed changed but that is expected for motors with finite torque.

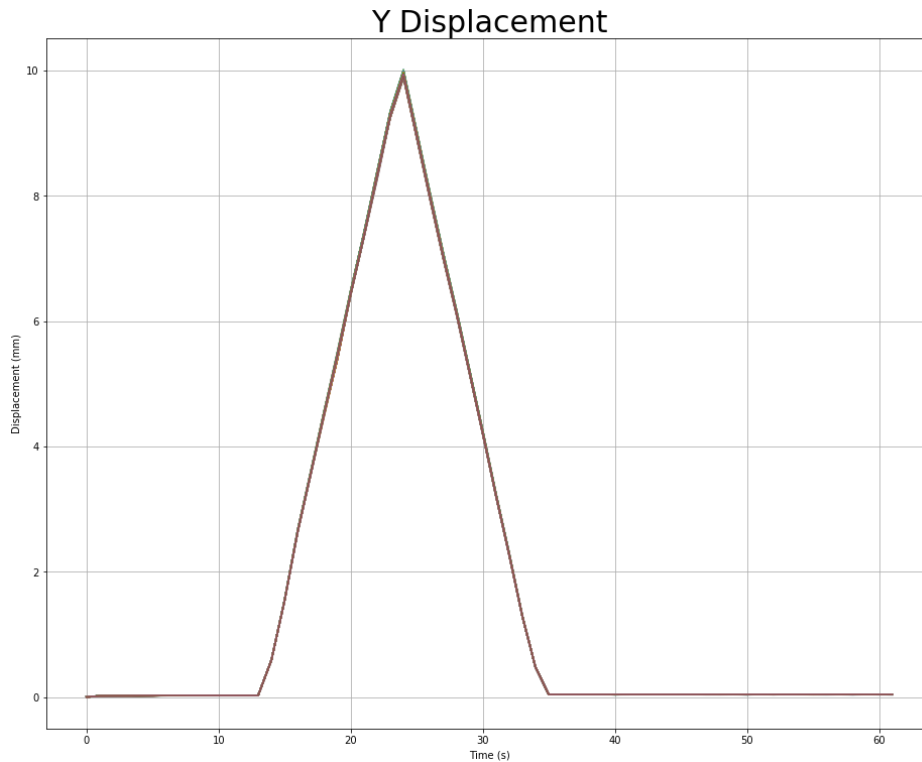


Figure 50: Y displacement measured by 2D DIC

The peak of the travel is at 24 seconds in the experiment. This peak was analysed for its accuracy to the 10mm displacement expected. The displacement at this peak is presented in Table 14 below:

	Subset X								Average Error
	1	2	3	4	5	6	7	8	
1	9.9769	9.9664	9.9499	9.9371	9.9241	9.9095	9.8964	9.8766	Target: 10mm
2	9.9871	9.9704	9.9591	9.9454	9.929	9.9129	9.8978	9.8899	
3	9.9835	9.9729	9.9592	9.9432	9.9322	9.9163	9.8999	9.886	
4	9.9865	9.9749	9.9635	9.9455	9.9353	9.9194	9.903	9.89	
5	9.9923	9.9807	9.9736	9.953	9.942	9.927	9.9127	9.8969	
6	9.9944	9.9827	9.9689	9.9533	9.942	9.9283	9.912	9.8927	
7	9.9997	9.9888	9.9699	9.9621	9.9465	9.9304	9.9199	9.8981	
8	10.004	9.9898	9.9777	9.962	9.9486	9.9363	9.925	9.9068	
9	10.002	9.9887	9.9766	9.9687	9.9496	9.9367	9.9226	9.9057	
10	10.01	9.9949	9.9849	9.9677	9.9543	9.9416	9.9278	9.9122	
11	10.014	9.9974	9.9845	9.971	9.9584	9.9464	9.9316	9.9129	
12	10.008	9.9967	9.9887	9.9741	9.9598	9.943	9.9294	9.9134	

Table 14: Peak Y Displacement for all Subsets

These results show a very accurate readings but the variation in values is higher than what would be expected from the noise floor experiments conducted earlier. There must, therefore, be other factors that are affecting the accuracy of the reading.

The main factors that would cause these discrepancies are mechanical tolerances. The ability of the test jig to reliably replicate a motion command is limited to an accuracy of (at worst) 0.08mm (see Table 13). Furthermore, Table 14 has been presented in the orientation of the subsets. Notice that the bottom left subset is linearly higher than the top right. Plotting the bottom left, top right and middle subset from time point 0 to the peak results in Figure 51. This clearly shows a trend that as the experiment advances the target starts tilting away from the

cameras. This is a result of the Y axis (as seen by the cameras) not being totally aligned with the motion of the target.

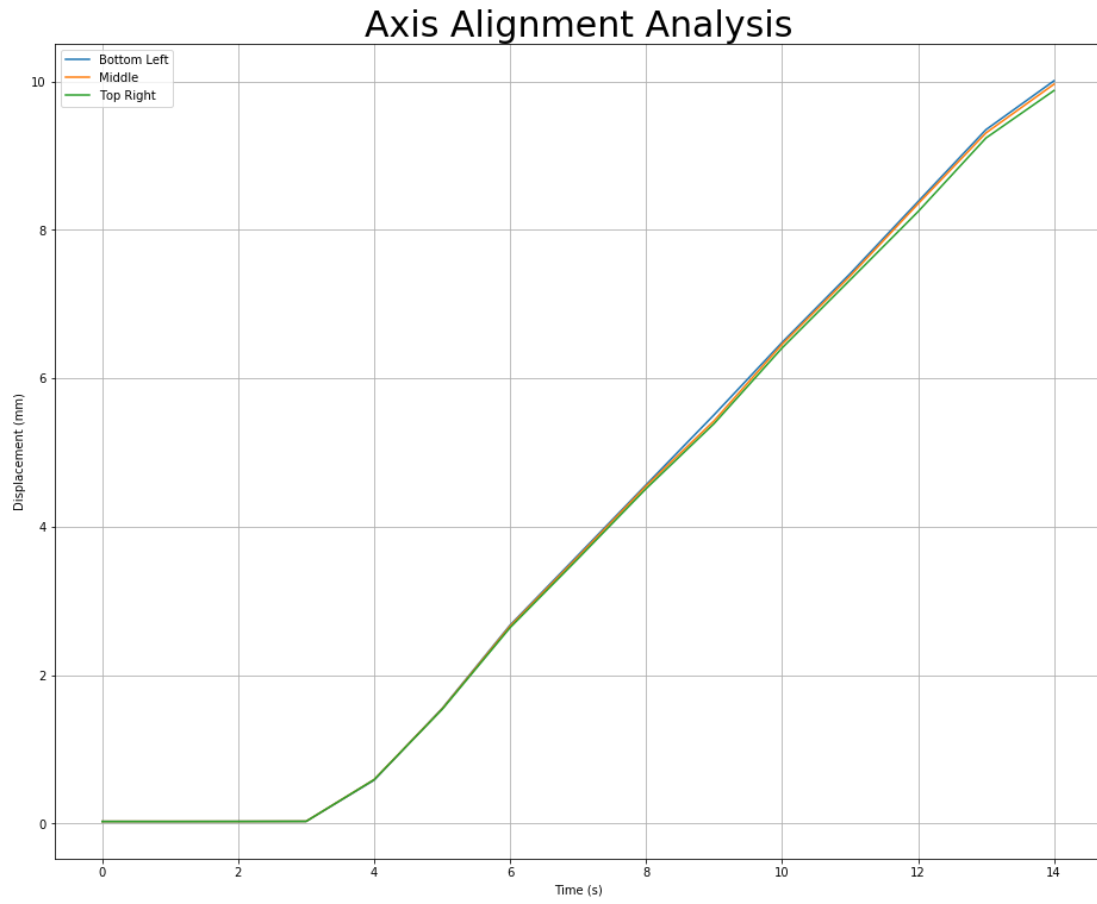


Figure 51: Axis Alignment Analysis.

4.4.4. Three-Dimensional Displacement - Stepped Profile

4.4.4.1. Procedure

The test jig was configured to measure displacement in Z. The target was a static speckled image which was mounted in the frame. The cameras were then pointed at the frame.

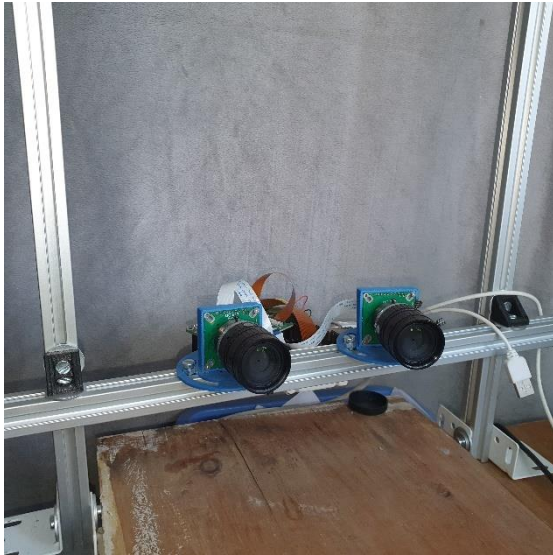


Figure 52: Displacement Measurement Equipment Setup

A script was run that stepped forward 1mm for four steps. This experiment was run twice with two different frames of reference. The point here was to show that the frames of reference were implemented in the system and to show what effect they had.

As part of this experiment a demonstration of the frames of reference built into DICE is also undertaken. Frames of reference can be important if the frame of reference for the measurement is not directly linked to either camera in the system. The frame of reference is stipulated with a file called *best_fit_plane.dat*. Here, the frames of reference chosen were:

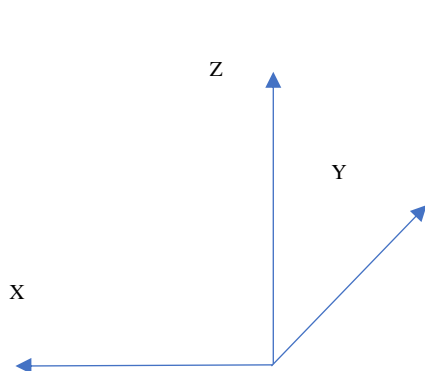


Figure 54: X/Y Negative Frame of Reference

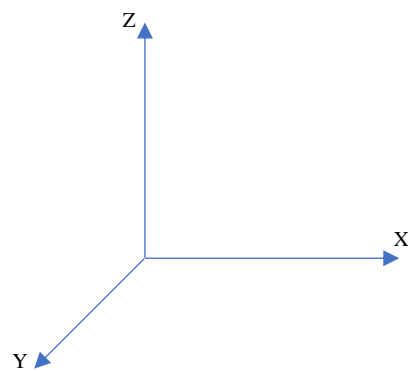


Figure 53: X/Y Positive Frame of Reference

4.4.4.2. Results

After running the script the Z displacement was captured, as shown in Figure 55.

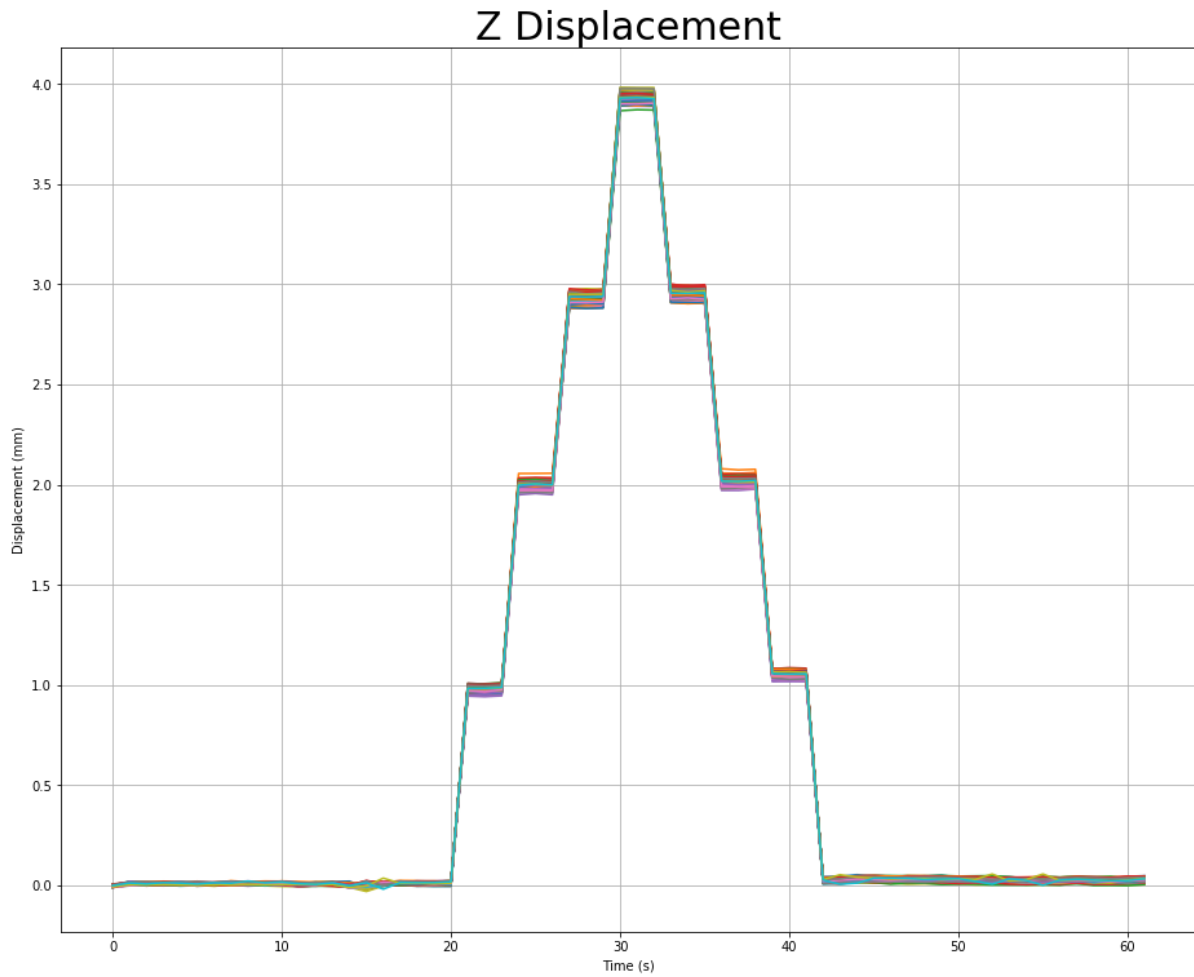


Figure 55: Z Displacement with Stepped Motion

The data at each step was sampled and analysed to extract how accurately this has been measured, with the detailed data in Appendix 2.

The average error at each step is summarised in Table 15.

Sample	Error	Standard Deviation
1mm	0.0217 mm	0.012
2mm	0.0127 mm	0.016
3mm	0.0694 mm	0.021
4mm	0.0658 mm	0.021
3mm	0.0486 mm	0.021
2mm	0.0191 mm	0.017
1mm	0.0525 mm	0.012

Table 15: Z Displacement Error Measurement.

Once again, the variation in reported displacement is higher than would be explained by simply the noise floor. This experiment relied on the test frame moving forward in small increments and then reversing the process. From the calibration of the test jig in Table 13 we determined that the test jig generally moves less than the requested command. When moving in small steps this command inaccuracy along with general backlash/play in the lead screw would explain the slight inaccuracies in the data reported vs the expectation.

If the explanation of these inaccuracies is indeed mechanical concerns, all subsets should report roughly the same values and the standard deviation should be within the noise floor. Table 15 includes the standard deviation of the measured values and in general the standard deviation is indeed close to the 0.017mm seen in the noise floor experiments. For the displacement at 3mm and 4mm the standard deviation is slightly higher, and this could be explained by axis misalignment with the axis of the cameras. One can see in Figure 55 the spread of the values increased as the displacement increases.

In Figure 56 one can see that the displacement in Z is positive but for the two different frames of reference the X and Y displacements are measured differently. This shows how the frames of reference have an effect on the measurements taken within the system.

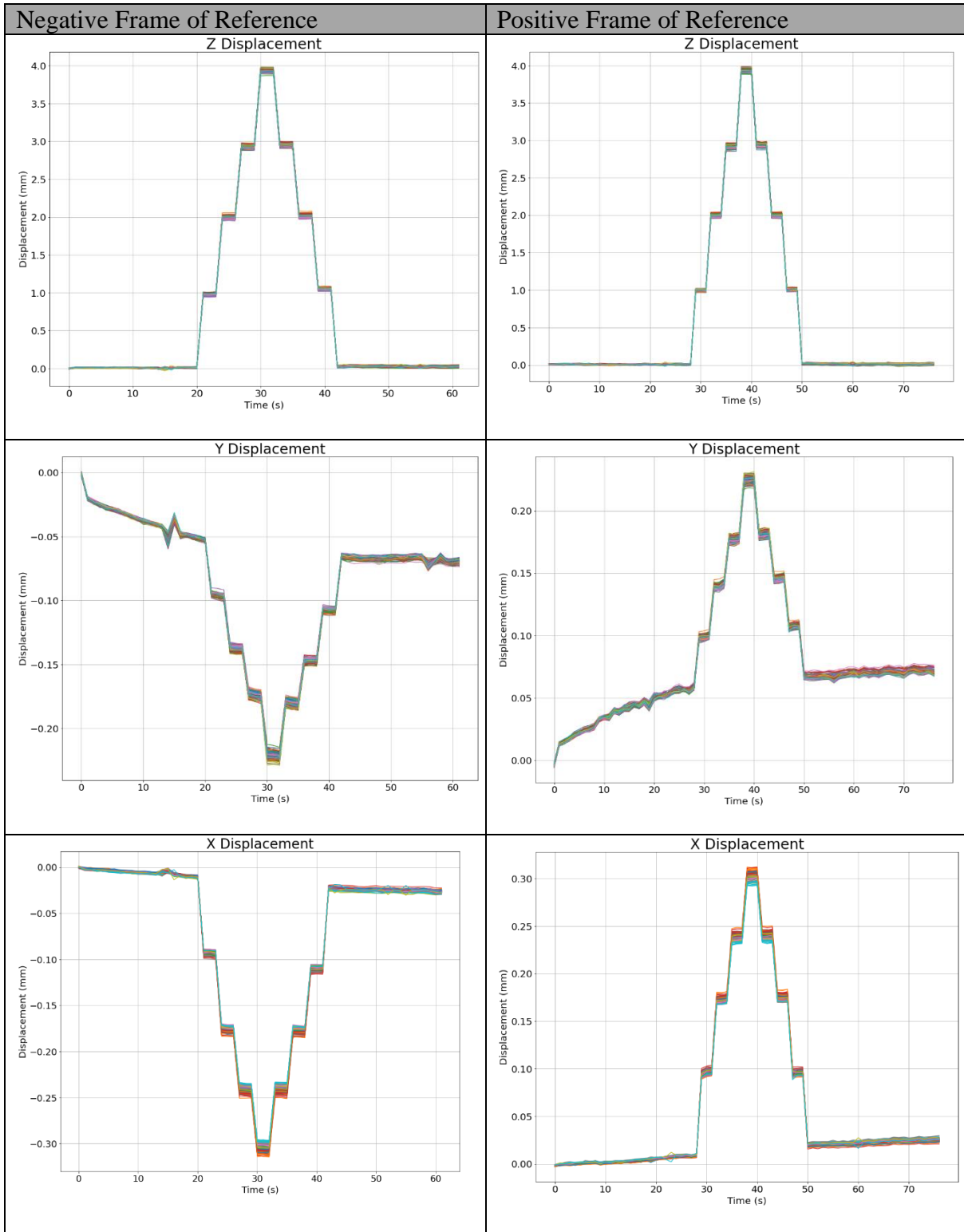


Figure 56: Positive and Negative Frame of Reference.

4.4.5. Three-Dimensional Displacement - Slow Continuous Profile

4.4.5.1. Procedure

The same script was run that was used in the experiment in Section 4.4.3. The cameras were now pointed forward as per the setup in the experiment in Section 4.4.4.

For this experiment 95 subsets were monitored.

4.4.5.2. Results

After running the script, the Z displacement was captured, as shown in Figure 57.



Figure 57: Z Displacement with continuous motion

The peak of the travel is at 25 seconds in the experiment. This peak was analysed for its accuracy to the 10mm displacement expected. The displacement at this peak is presented in Table 16.

Subset Y	Subsets X								Average Error
	1	2	3	4	5	6	7	8	
1	10.229	10.193	10.229	10.212	10.184	10.169	10.165	10.132	
2	10.245	10.184	10.231	10.188	10.188	10.173	10.159	10.256	
3	10.19	10.2	10.207	10.17	10.16	10.169	10.144	10.25	Target:
4	10.194	10.22	10.205	10.162	10.161	10.203	10.15	10.246	10mm
5	10.201	10.211	10.189	10.191	10.194	10.148	10.161	10.245	
6	10.187	10.203	10.199	10.188	10.186	10.193	10.155	10.236	
7	10.198	10.228	10.207	10.193	10.158	10.15	10.171	10.224	
8	10.191	10.204	10.169	10.207	10.145	10.146	10.155	10.251	
9	10.214	10.221	10.189	10.194	10.155	10.16	10.173	10.252	Variance:
10	10.188	10.194	10.177	10.204	10.161	10.177	10.166	10.257	0.18mm
11	10.192	10.208	10.173	10.222	10.174	10.17	10.165	10.248	
12	10.167	10.215	10.166	10.192	10.181	10.155	10.147	-0.01181	

Table 16: Z Displacement Error Measurement.

The script run for the slow continuous movement up to 10mm and back obviously explores different mechanical aspects of the system. From Table 16, it is obvious that the command for a single step of 10mm resulted in an overshoot of approximately 0.18mm. In comparison, the stop-start nature of the tests in Section 4.4.4 resulted in displacements slightly less than that commanded. This supports the hypothesis that the inaccuracies in frame motion were due to mechanical play.

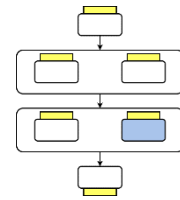
It is also clear that the subset at row 12 column 8 did not track. For the calculation of error variance this was excluded.

4.5. Experiment 5 – Motion (Speed)

4.5.1. Description

The most basic metric of a real time capture of data will always be the sample rate. There will always be a sample rate at which the actual meaning or the ability to reconstruct the fundamental data starts being lost.

In digital systems the sample rate required to accurately reconstruct the signal is referred to as the Nyquist limit. In these experiments we are trying to understand what limitations exist in the system with regards to sample rate such that we can still reconstruct the displacement data in the experiment in sufficient detail.



Data Set	Description
Speed/speed_1	0.8mm/sec data
Speed/speed_2	1.4mm/sec data
Speed/speed_3	5mm/sec data
Speed/speed_4	10mm/sec data

Table 17: Experiment 5 Data Sets.

4.5.2. Procedure

The same script was run that was used in the experiment in Section 4.4.3. Each time the script was run the speed of the motion was increased. In each case the system was allowed to freely sample the Z displacement (i.e., sampling was as fast as the system would allow).

4.5.3. Results

The free sampling approach used means that we need to extract the sample rate from the timing file generated with each experiment. For each of the tests below we expect the average sample rate to be roughly the same as the number and size of the subsets is constant. The only difference should be the speed of the motion.

The sample rates of the experiments are given in Table 18.

Experiment	Sample Rate
0.8mm/sec	1.44599585s
1.4mm/sec	1.48940113s
5mm/sec	1.41003483s
10mm/sec	1.4147066s

Table 18: Sample rates at varying speeds.

As expected, the sample rate of the experiments was consistent across experiments. The variability of the sample frequency across the standard motion profile is now shown at various speeds.

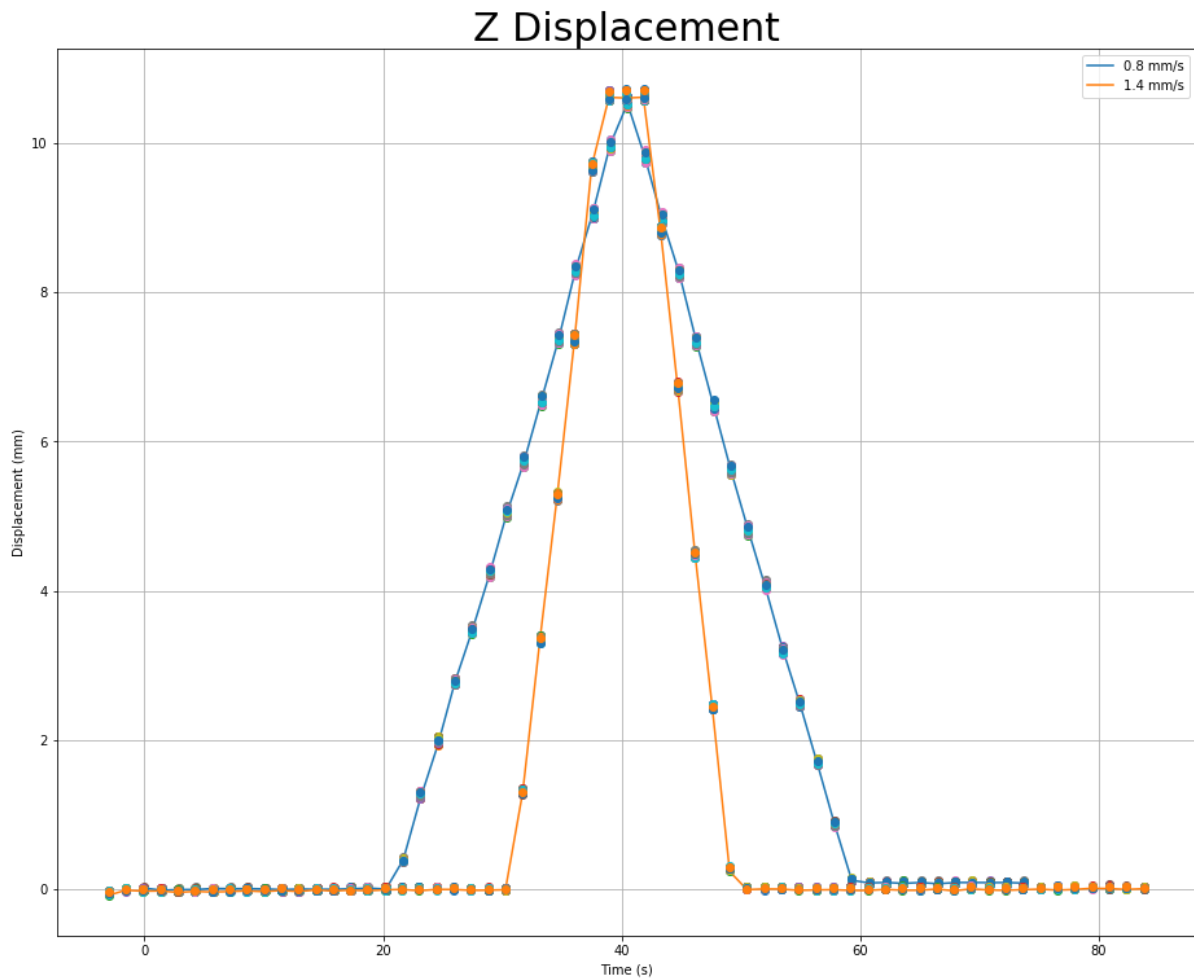


Figure 58: Speed set at 0.8mm/s and 1.4mm/s.

In Figure 58 the profile captured at 0.8mm/s reconstructs the profile of the motion accurately. The speed of 1.4mm/s also reconstructs the profile, but one can see that the sample rate is lower and we are in danger of losing details if there was some unexpected (non-linear behaviour) between samples. In Figure 59 the data for 5mm/s is shown. While the peak displacement is approximately correct, it is clear we have not captured the shape of the displacement history acceptably. We have no way to ascertain the profile except to linearly interpolate which could be very inaccurate.

The results presented here are specific to this type of motion profile. Varying the motion profile will change what the Nyquist limit for the experiment is. In this case the maximum speed that that acceptably reconstructs the motion is 1.4mm/sec. At 5mm/sec it is clear that the motion is not captured – only the start and stop points are sampled.

If the goal of the experiment is accuracy this would suggest that a speed of 1mm/s or less would be suggested.

Z Displacement

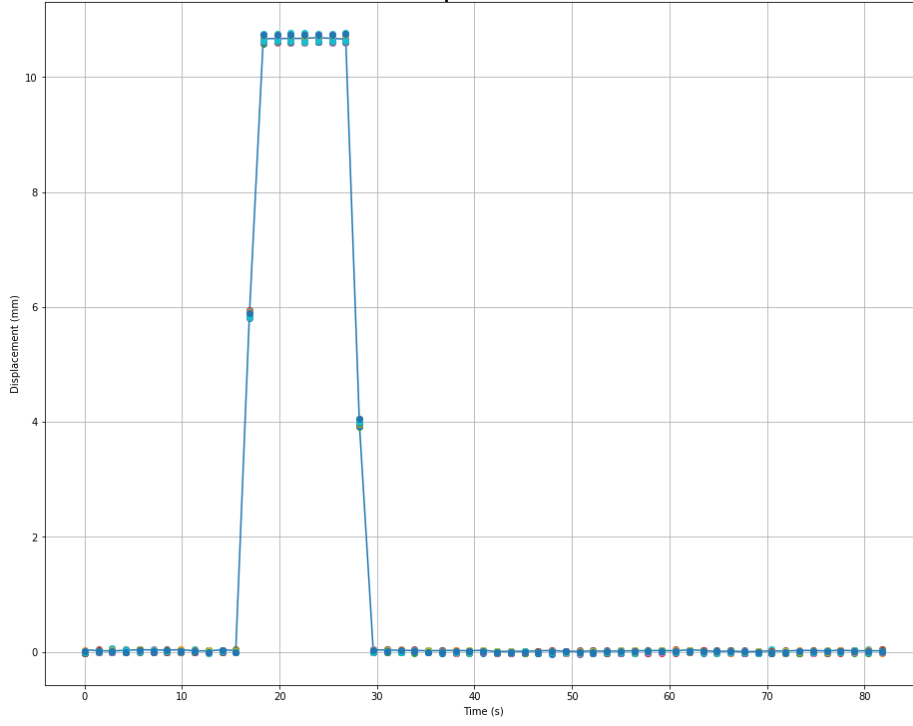


Figure 59: Speed set at 5mm/s.

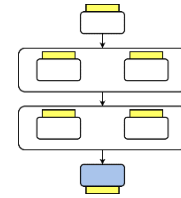
4.6. Experiment 6 – Bulge Test

4.6.1. Description

For this experiment the setup, as shown in Figure 28 was used. Two indenters were used:

- a small, 25mm indenter
- a larger 40mm version.

The indenter is aluminium, and CNC machined to a known radius. It is effectively rigid in comparison to the latex sheets used in this experiment which means the latex will deform to this known radius where it contacts the indenter. These sizes were chosen since they represent the typical range of diameters in biological bulge test research done previously [82].



The frame was moved backwards over the indenter with the latex surface in place. This had the effect of causing the latex to bulge much as it would in a traditional bulge test experiment.

Data Set	Description
Bulge/indent_small	25mm indenter experiment
Bulge/indent_big	40mm indenter experiment.

4.6.2. Procedure

- Calibrate the stereo camera set.
- Install the 25mm/40mm diameter indenter into the system. See Figure 60.
- Create a subset input file that covers the region that the indenter would bulge. The subset size for both experiments below was set to 30 x 30 pixels.
- Position the frame such that the latex surface in the frame is touching the indenter but no physical indentation is happening.
- Start the DIC algorithm.
- Move the frame 10mm backwards at a very slow speed (a script was used for this).
- Extract the MODEL_DISPLACEMENT_Z results for each subset from the DICe solutions file.
- Extract the MODEL_DISPLACEMENT_Y results for each subset from the DICe solutions file.
- Extract the MODEL_DISPLACEMENT_X results for each subset from the DICe solutions file.



Figure 60: 25mm indenter installed into system.

The indenter setup results in the latex surface bulging and following the profile of the indenter. Figure 61 shows the nominal end of experiment setup.

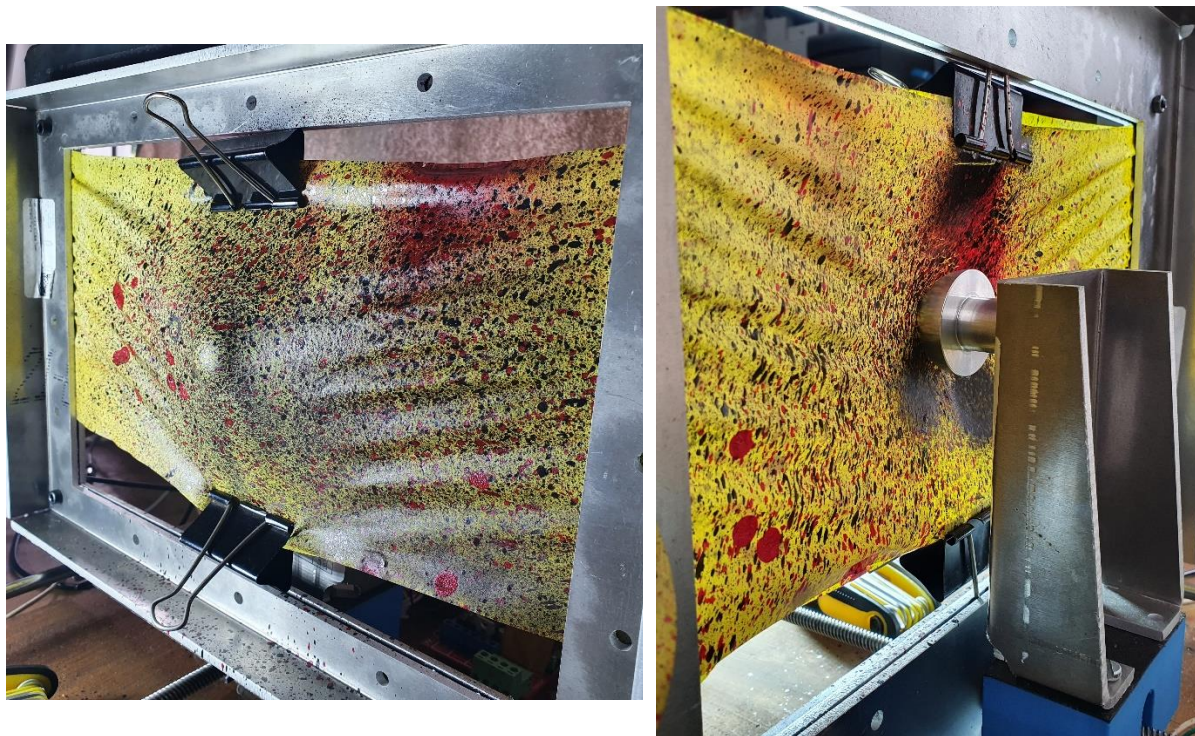


Figure 61: Latex bulge due to indenter.

In order to analyse the bulge for accuracy a least squares method was used. This method was used to numerically estimate the radius of the bulge. As shown in Figure 62, the subset chosen for tracking here does complicate this approach. If the red subset in Figure 62 was chosen, the estimated radius could lie anywhere between the red and green circles. Therefore, a smaller subset of data is needed for accurate radius estimation, but means fewer data points are available.

For analysis, a subset of the sampled data was extracted that best approximated where the latex followed the curve of the indenter. This was then numerically analysed to illustrate if the DIC method used could faithfully measure the bulge.

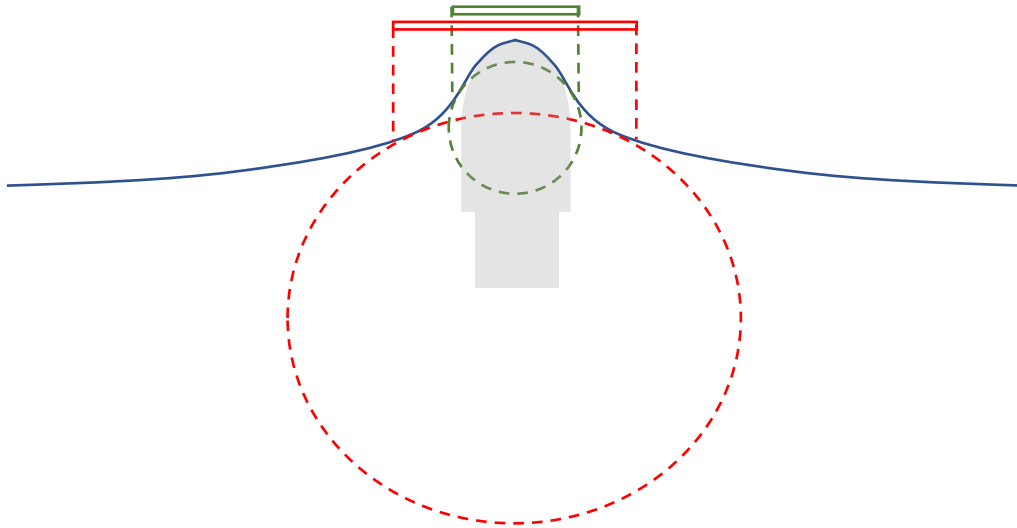


Figure 62: Illustration of the Estimation of the Curve.

4.6.1. Results for larger indenter

A sample of the real time heat map images are presented in Table 19.

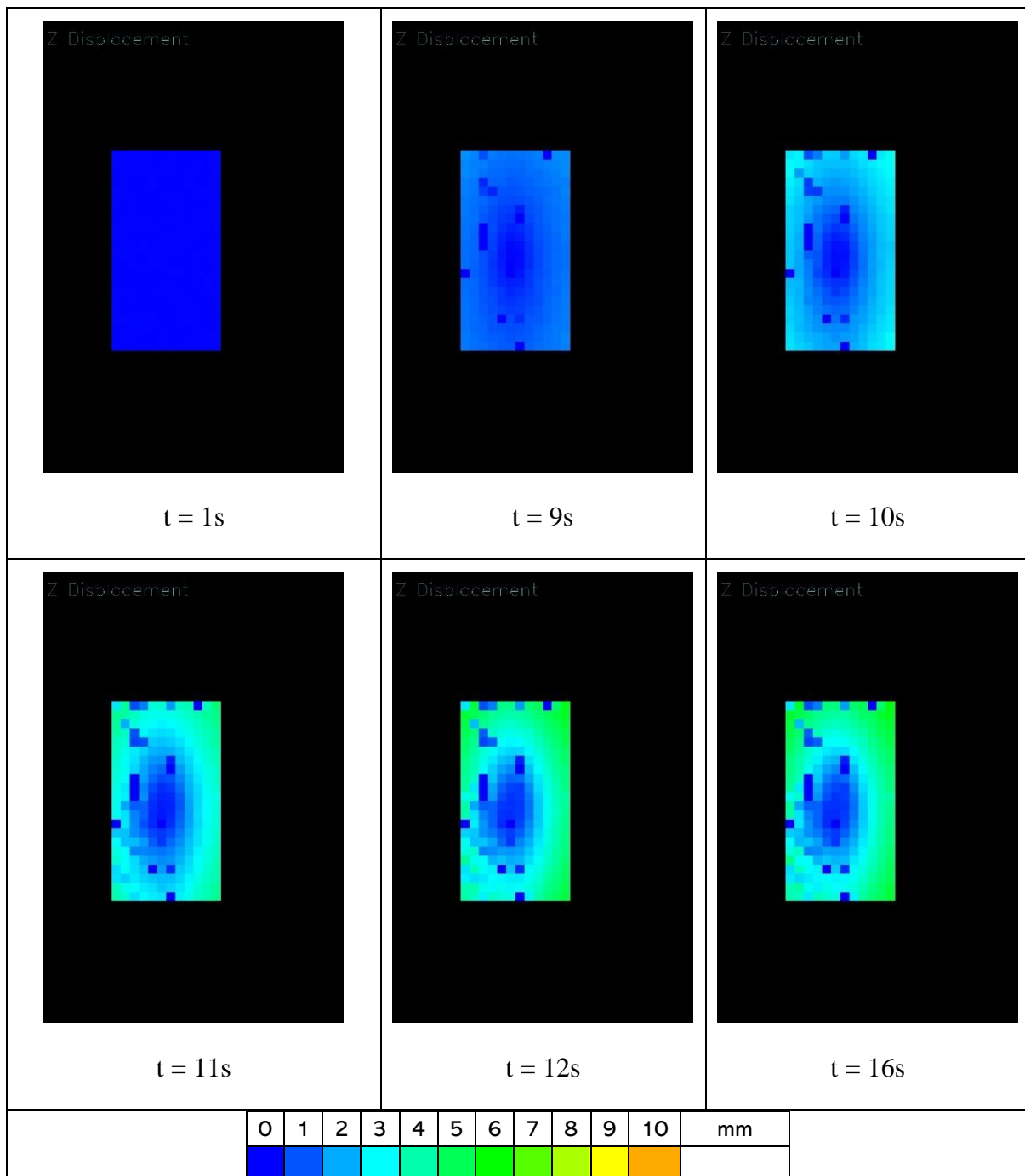


Table 19: Real time displacement images (large indenter).

These images were displayed in real time as the test was being executed. As can be seen, certain subsets never moved from the original blue. This was due to those particular subsets not having enough features to track for the duration of the experiment.

Since the latex sheet was moving back over the indenter the displacement was in a negative direction with regard to the cameras. Hence in the heat map, the bluer the subset the closer to the cameras and the greener/more orange the further the target is from the cameras.

As mentioned, there are some subsets that are not trackable since there were not enough features identified. This is the result of the density and feature profile of the speckle pattern in that area and not the DIC algorithm itself.

In order to display the bulge effectively the non-trackable points have been excluded.

An example of the raw versus filtered data is shown below to demonstrate that the representation of the data is still true. This filtering was carried out as a post process on the data.

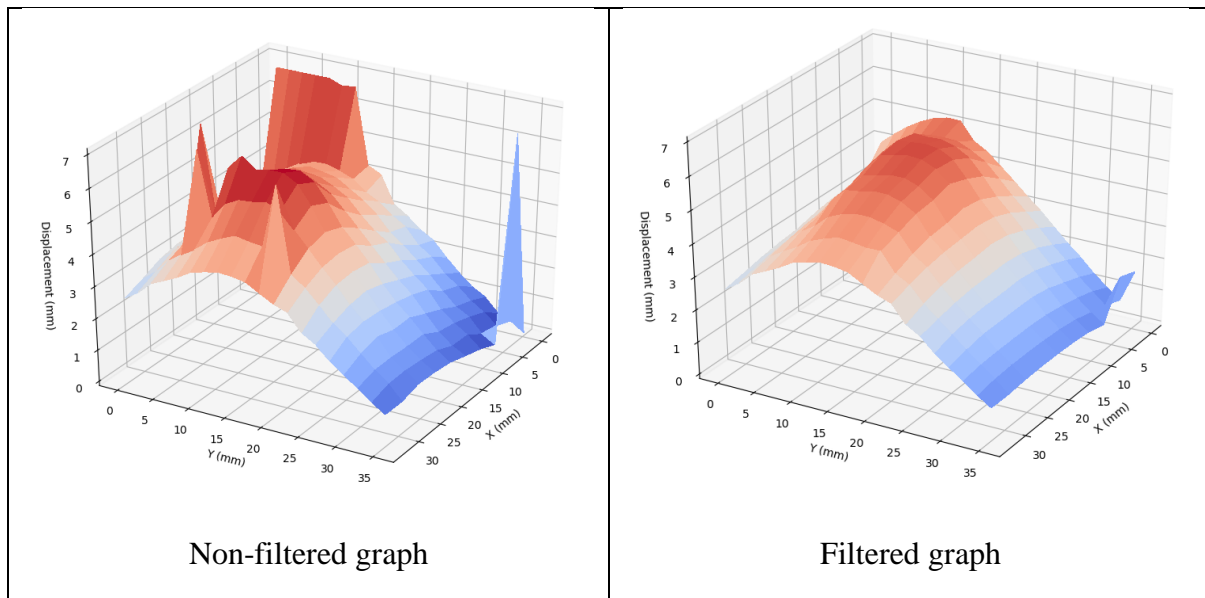


Figure 63: Example of the filtering algorithm.

In Figure 64 the progression of the bulge test is illustrated with graphs plotted from the results files. One can clearly see the convex sphere of the indenter become apparent as the experiment continues.

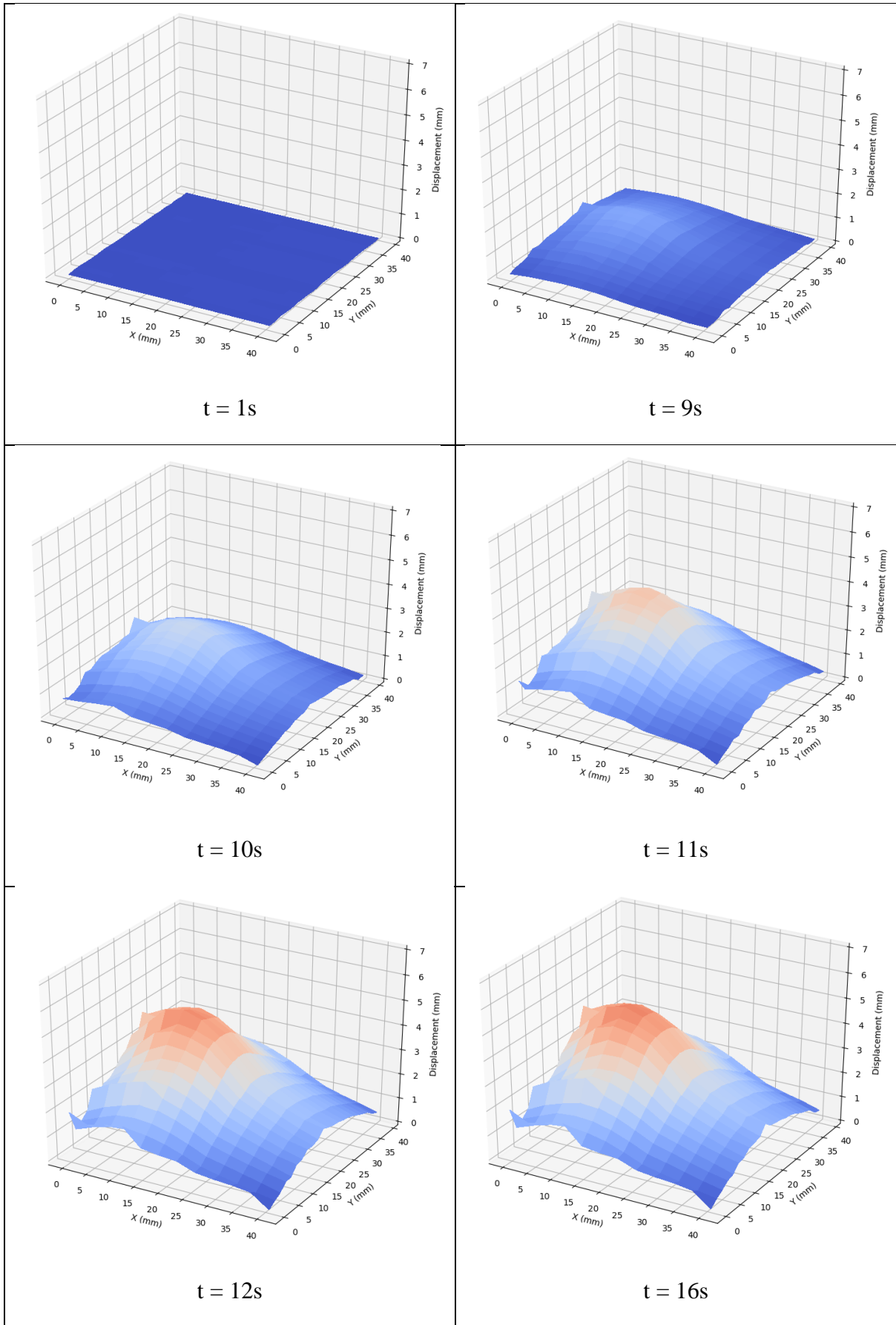


Figure 64: Progression of Bulge Test.

A numerical method was used to extract the radius of the bulge as measured by the DIC algorithm [83].

First a set of the final data that represented the actual bulge impression was chosen. This is shown in Figure 65.

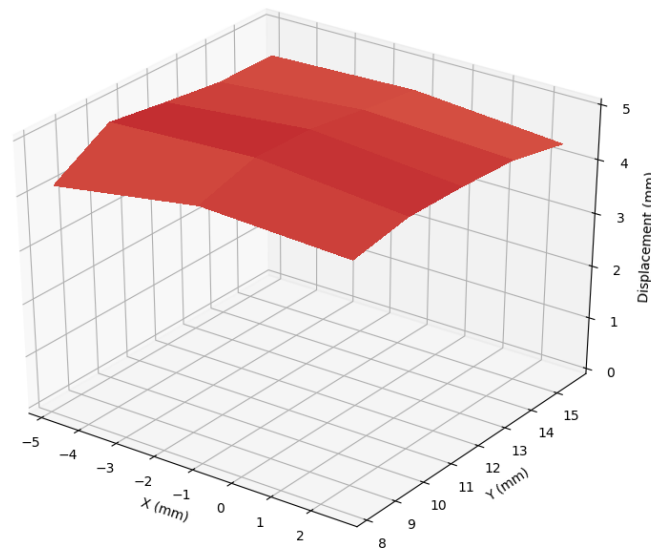


Figure 65: Extract of the data representing the bulge – large indenter.

This set of data was used in the least squares algorithm to extract an approximation of the sphere that would have created this bulge. The data points for the sample used here were graphed and the approximate sphere was also plotted. The result is given in Figure 66.

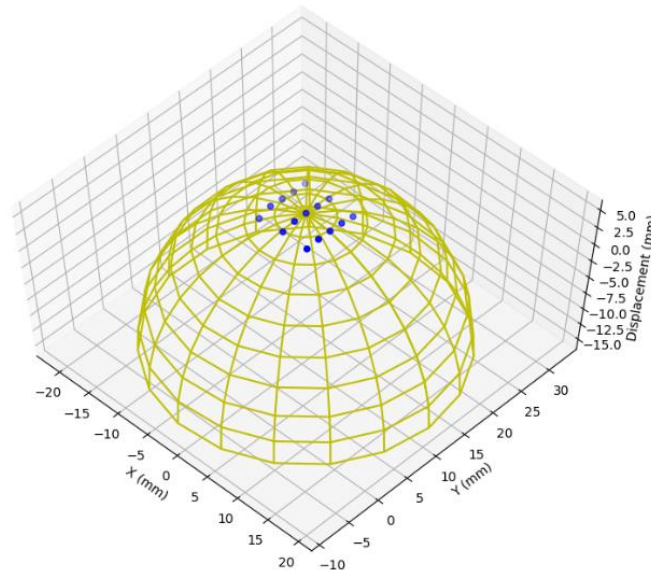


Figure 66: Bulge estimation – large indenter.

For this data set a radius of 20.2mm was calculated from the data points. The indenter radius is CNC machined to 20.0mm, with an expected tolerance of 0.02mm. The 3D DIC deformation measurement was therefore accurate to within 1%. Although not officially

considered in these results, the thickness of the latex sheet could mean that the result is more accurate than this. Assuming the stretched latex sheet has a thickness of roughly 0.1mm it would mean the actual indenter is being measured at less than 20.2mm which would mean the system is accurate to less than a 1% tolerance.

4.6.2. Results for smaller indenter

For the smaller indenter a sample of the real-time heat map images are presented in Table 20.

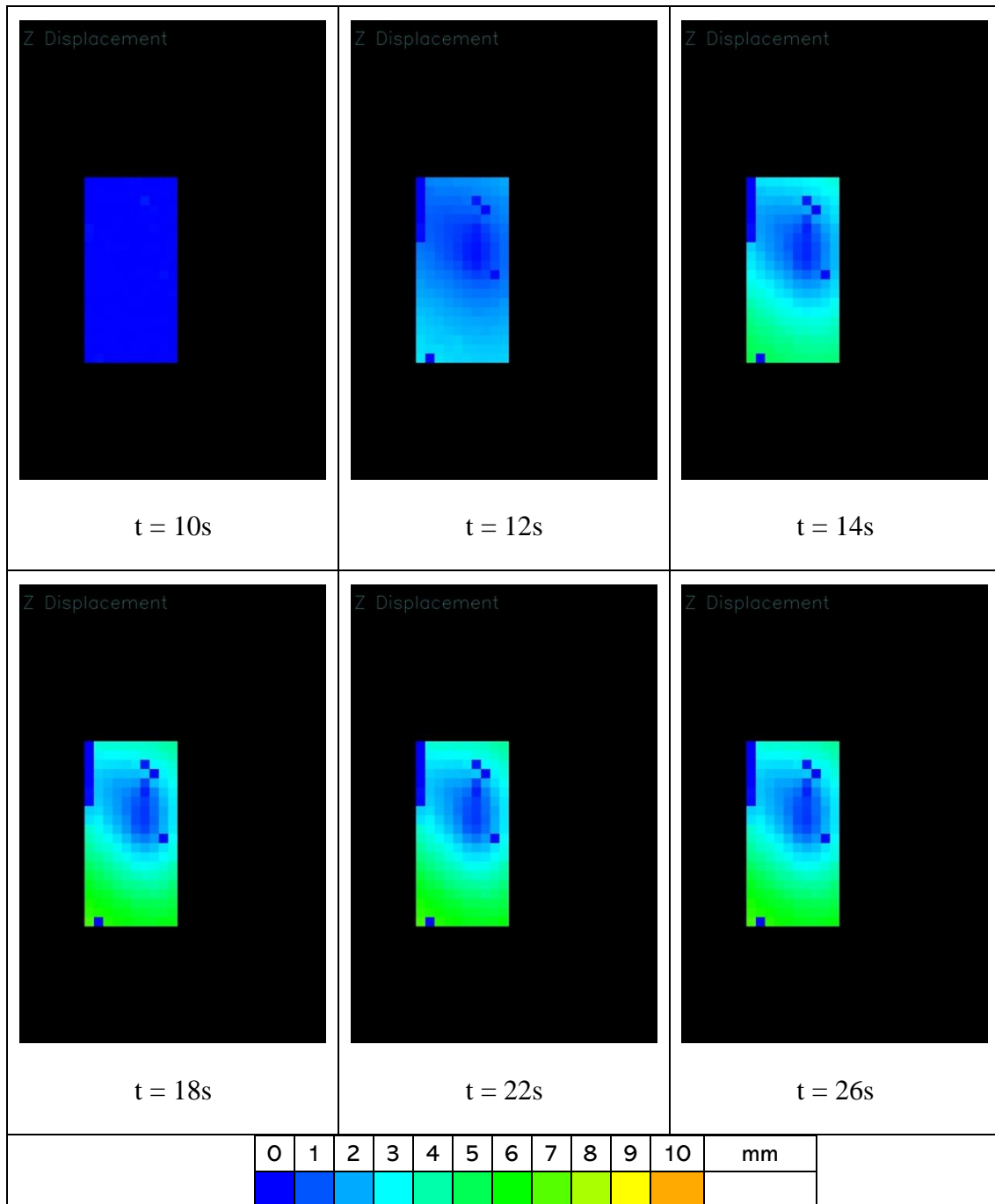


Table 20: Real time displacement images (small indenter).

These images were displayed in real time as the test was being executed. For this experiment a smaller region of interest was chosen since the indenter was smaller.

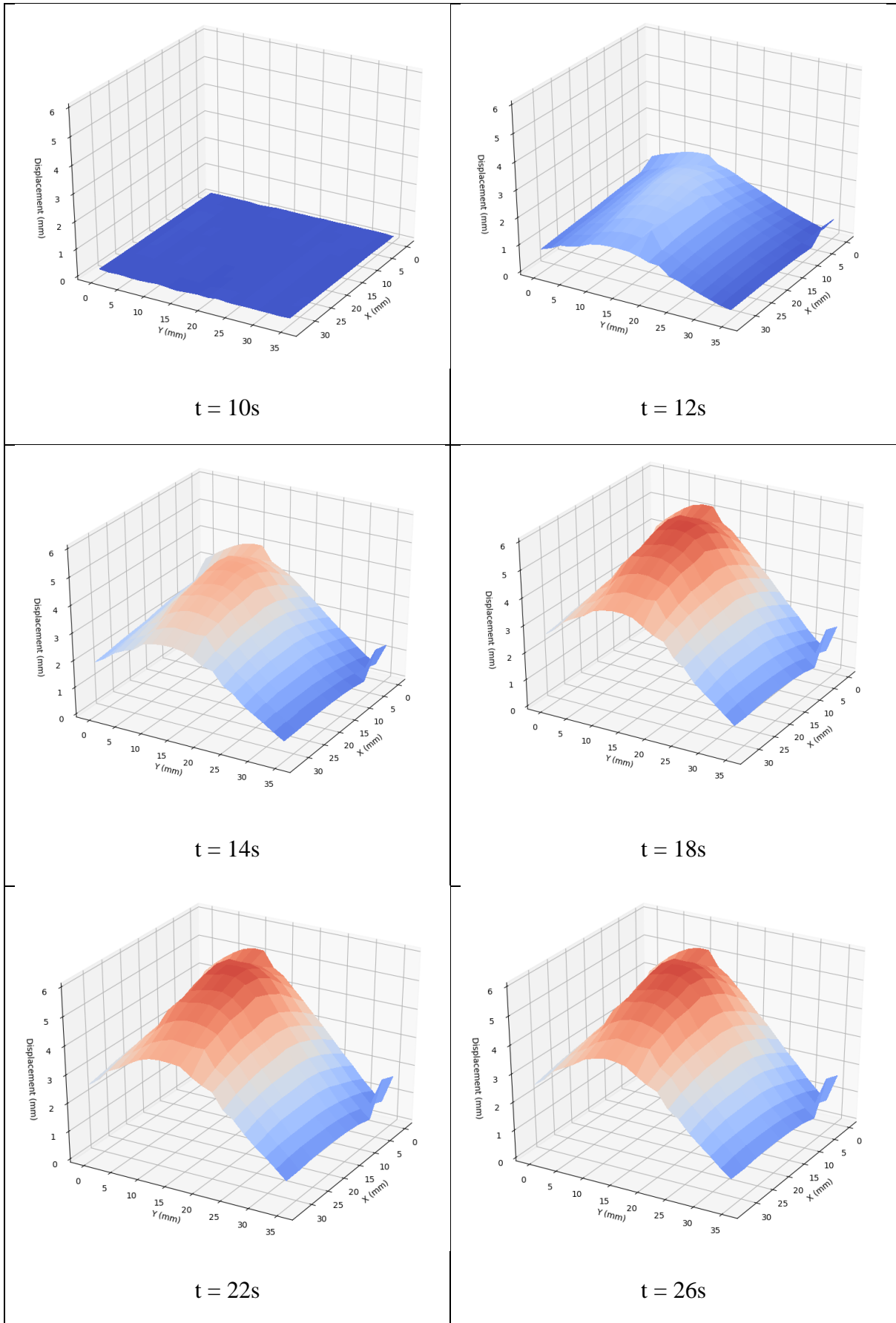


Figure 67: Progression of Bulge Test.

Figure 67 represents the same data as Table 20 but extracted as part of the post processing. Using the same algorithm as used previously, the non-trackable subsets have been filtered out.

As done previously, a set of the data was chosen that best represented the bulge, and this was run through a least squares algorithm to determine the radius. The subset chosen is shown in Figure 68.

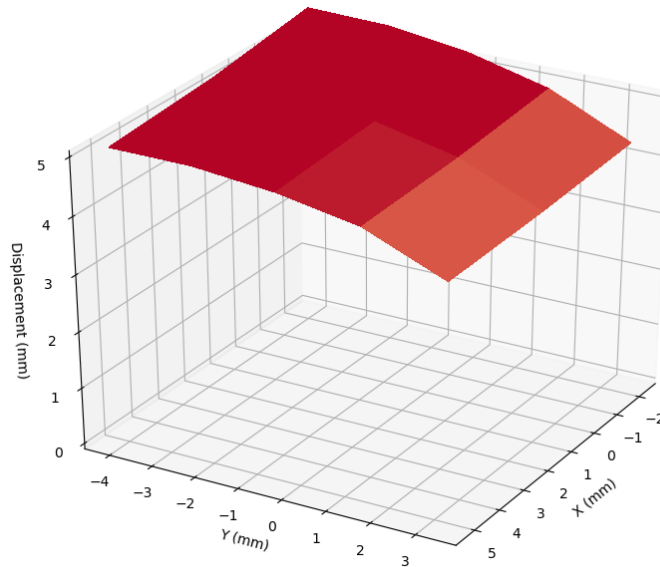


Figure 68: Extract of the data representing the bulge – small indenter.

The radius was then estimated as shown in Figure 69.

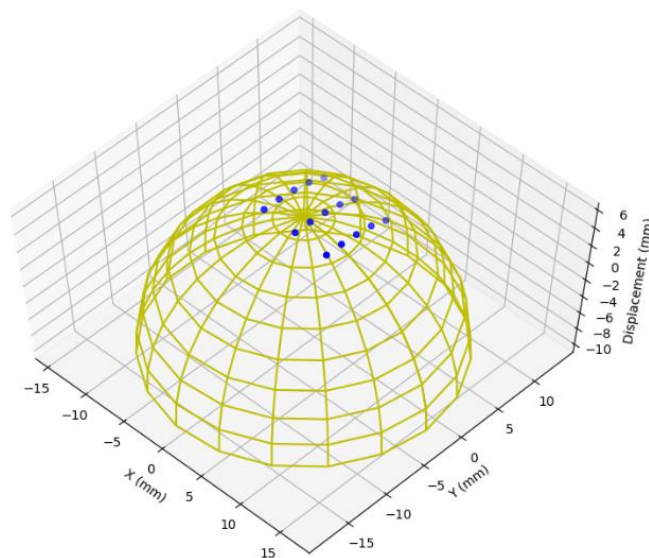


Figure 69: Bulge estimation – small indenter.

For this set of data, a radius of 15.5mm was calculated from the data points. The physical measured (and machined) radius is 12.5mm . This represents a much larger error than the previous experiment.

This experiment highlights the degree to which the subset size needs to be considered in the execution of an experiment. With a larger sample (large indenter) the larger subset size had less of a negative effect on the accuracy. This is since the larger the sample the better the larger subsets can more accurately approximate the curve.

With the smaller indenter the larger subsets cannot match the surface accurately which means that the estimate of the radius is exaggerated (much like the illustration in Figure 62).

To test this the same experiment was run with subsets half the size. This results in a lot more points that cannot be tracked since the speckle pattern used did not have the granularity to allow for tracking at this subset size. The tracking inconsistencies can be seen in Figure 70.

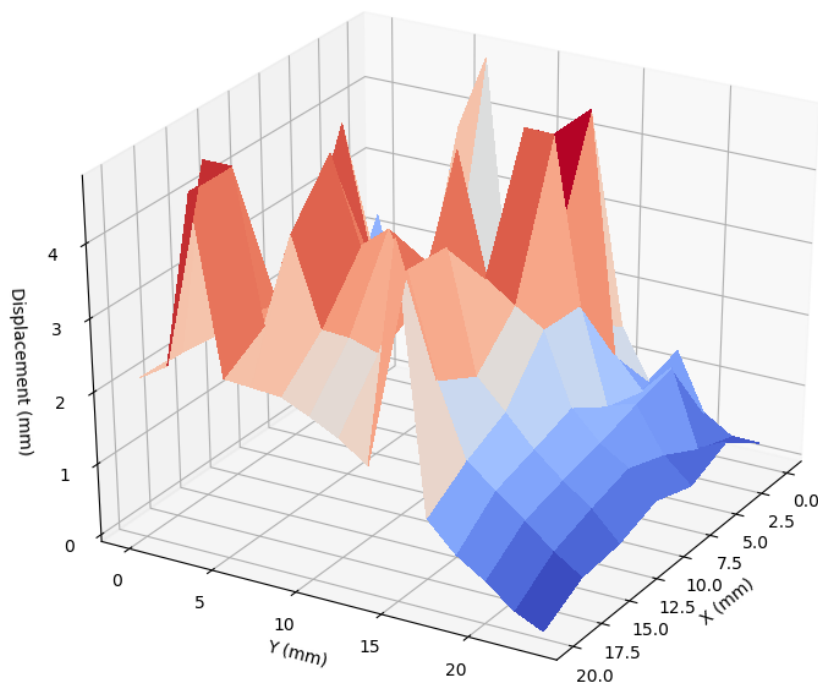


Figure 70: Smaller Subset Size Tracking

In the interests of understanding if this would result in a more accurate representation of the sphere the data was filtered, and the radius of the sphere estimated using the same method above. After this processing the radius was shown to be 11.5mm . This can be seen in Figure 71. Although this matches the expected radius much better the level of processing that was needed on the input data is not practical. For this subset size to be used a much better process for speckle pattern creation would need to be used.

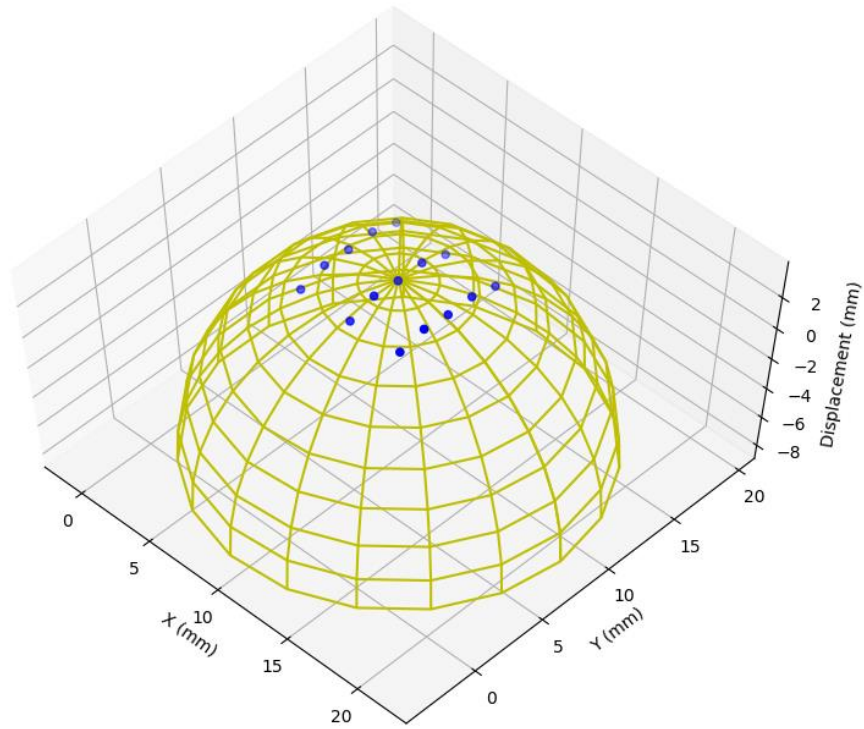


Figure 71: Estimate of radius with smaller subsets used.

For accurate measurements, therefore, a well-considered (and in most cases small) subset size would be required. A smaller subset size for the same region of interest would mean more subsets which we showed definitively in section 4.3 will not compromise the near real-time nature of the results.

Smaller subsets sizes do however require a much finer pitch of speckles since each subset would need to present as a unique specimen in the correlation.

5. Concluding Discussion

5.1. Feasibility Discussion

5.1.1. Summary

We set out to understand to what extent it was practical to run 3D DIC algorithms in real-time on a single board computer. The question arose from the following:

- In bulge test experiments conducted with softer, fragile materials, a real time measure of deformation or strain is beneficial for interrupting experiments before rupture.
- At this point DIC is generally run as a post processing activity on a standard computer – usually one with significant specifications. Given the advances in the size and power of single board computers the DIC algorithm could potentially run on a single board computer in real time. This would increase the portability of the DIC system.

Based on the code bases' open source status and its modular nature the DICe code base was chosen for the implementation of this project. The NVIDIA Jetson Nano single board computer was also chosen based on its availability, cost effectiveness and capabilities.

The DICe code base then had to be compiled on the Jetson Nano single board computer and an application then had to be developed around this code base.

With the application developed, a test environment was created to test the efficacy of the code base to answer the questions as mentioned at the start of this paragraph.

5.1.2. Results Summary

The noise floor experiments showed that the system produced has low noise characteristics. Any experiment where the measurement required needs a resolution $\geq 0.02\text{mm}$ will not be affected by the noise. It is important to note that this result does not mean that the system is accurate to this resolution, only that this is the noise floor of the system. Anything that tries to measure at an accuracy below this will, by definition, be inaccurate.

The test jig's motion was confirmed to be accurate to within 0.1mm. While it would be helpful to have a more accurate reference for motion, due to travel and laboratory restrictions associated with the COVID 19 pandemic, it was not possible to access a more accurate test frame within the timelines of this project.

Since measuring the 3D displacement via DIC is based on measuring the 2D displacement in both cameras, accuracy of the 2D displacement measurement was checked first. This was found to be accurate to 0.1mm (on average).

The 3D displacement measurement accuracy was measured using a stepped profile and a continuous profile. The final accuracy measured was (on average) 0.1mm.

All of these results showed the system can execute the 3D DIC algorithm in real-time, while producing acceptable accuracy. Due to mechanical play in the test frame, it wasn't possible to confirm if the DIC was achieving better accuracy. In order to show the real-time possibilities of the system the speed at which the system can execute the algorithm needed to be tested.

The subsequent experiment determined the effect of the number of subsets on the speed of the execution. The rule of thumb was determined to be that ~ 0.04 sec added to the processing time per subset added. This is however not a linear relationship. As shown in the experiment due to the nature of the processor used, the execution time plateaus at points where the multiple cores start getting used across multiple subsets. As the cores are fully loaded the execution time jumps to allow for the next set to be processed.

The actual speed of execution was measured but the interpretation of the result is harder to contextualise. What we were trying to do is determine what the profile of the displacement looked like as the sample was deformed. If only the start and stop positions of the sample were of interest, the actual speed of execution would be almost irrelevant since these could be sampled at any point. However, this is most likely to not be the case. For the experiment done, the processing speed achieved was roughly 0.8Hz. At this sample rate a speed of 0.5mm/s to 1.5 mm/s created a useable reconstruction of the waveform.

All of these experiments were done using a flat, rigid target. To prove that this method would (after all of these basic metrics were defined) be able to properly measure a bulge test this was simulated by deforming a flexible target, using two different radius indenters. Within the limits of the equipment available, the accuracy of the displacement profile measured for the larger indenter was shown to be accurate to $< 1\%$ of the actual radius. For the smaller indenter the interplay between speckle and subset size was a limiting factor.

5.1.3. Comments on Feasibility

The result of this set of experiments is that the concept of real-time 3D DIC on a portable, single board computer is feasible. This broad statement does however come with some caveats.

The processing power of the Jetson Nano (specifically) limits the system to completely processing displacement for one frame every 1.4s. A lower number of subsets can improve this but for all practical purposes the sampling rate is a frame every second. This means that any experiment where sampling the displacement at a faster rate is important, will make this implementation unsuitable.

The very high pixel density of the 12MP cameras used was unnecessary for this application. Faster sample rates could be achieved by using cameras with lower pixel density, but this would need to be carefully weighed up against the compromises in accuracy that would be inevitable.

Due to this limitation on speed, larger subsets are preferable to smaller ones on the sample. This means that the results will be more granular the larger the sample is – if one is still trying to achieve the real-time functionality. A smaller sample is therefore preferable.

In the process of executing these experiments, the importance of the target speckling quality became important. In regular DIC, where results are post-processed, a problem with speckling would be discovered after the experiment has concluded. The reason one might want a real-time DIC process is often because the sample is fragile and hence re-executing the tests may not be possible. Processes for better speckling and feature creation on targets are outside the

scope of this discussion but their importance became very clear. For this real-time process to work well the method of speckling needs to be deterministic in distribution and speckle size.

5.2. Potential Further Work

There are two main areas where further work could be conducted:

- Improving algorithm efficiency.
- Improving sample preparation.

In the area of algorithm efficiency there are various ways that improvements could be explored. The Jetson Nano is equipped with 128 NVIDIA CUDA cores. These allow for massive parallelisation of processes. The code as contained in the DICe library did not allow for easy translation to CUDA code, but this approach would offer significant performance benefits. Some work has been done on a CUDA code implementation of DIC algorithm but this would need to be translated to work with the DICe code base [73]–[76], [84].

There is also a possibility of improving efficiency in the DICe code base itself. The code base is optimised for standard DIC processes which are run as post-processes. For real-time applications some of the internals of the system could be improved/optimised. This was not explored but there is scope for this.

The sample preparation process has some scope for improvement as well. In the process of running the experiments it was found that the requirements for the speckle pattern distribution and randomness was directly related to the subset size. The relationship was not explored but it is an area that needs to be investigated further. Some work does exist on speckles and how to use them but this was not explored in great detail in this project [25], [42], [45], [47]–[49], [85]. Along with this, the actual fabrication of speckles is something that needs to be more deterministic. A summary of the speckle pattern fabrication techniques does exist, but many of the suggestions for speckle pattern fabrication required special equipment [8]. These fabrication methods need to be explored in the context of this work.

5.3. Concluding Remarks

In the introduction it was discussed that the vast improvement in computing power and availability of off the shelf components was the instigator of the question posed in this project. Ultimately, we proved that it is feasible, within certain constraints, to perform real-time 3D DIC with off the shelf components.

As technologies improve this will continue to be the case. During the course of this project, an updated Jetson Nano was released with 2 MIPI CSI-2 camera interfaces - meaning potential for a faster system is already available.

Ultimately improvements in camera technology, processor efficiency, calibration processes and speckle pattern application will make the entire exercise of 3D DIC very “plug and play”. The hope is that this project provides a useful step in that process.

6. References

- [1] G. Machado, D. Favier, and G. Chagnon, “Membrane Curvatures and Stress-strain Full Fields of Axisymmetric Bulge Tests from 3D-DIC Measurements. Theory and Validation on Virtual and Experimental results,” *Experimental Mechanics*, vol. 52, no. 7, pp. 865–880, Sep. 2012, doi: 10.1007/s11340-011-9571-3.
- [2] T. C. Chu, W. F. Ranson, and M. A. Sutton, “Applications of digital-image-correlation techniques to experimental mechanics,” *Experimental Mechanics*, vol. 25, no. 3, pp. 232–244, Sep. 1985, doi: 10.1007/BF02325092.
- [3] M. Sutton, W. Wolters, W. Peters, W. Ranson, and S. McNeill, “Determination of displacements using an improved digital correlation method,” *Image and Vision Computing*, vol. 1, no. 3, pp. 133–139, 1983, doi: 10.1016/0262-8856(83)90064-1.
- [4] H. A. Bruck, S. R. McNeill, M. A. Sutton, and W. H. Peters, “Digital image correlation using Newton-Raphson method of partial differential correction,” *Experimental Mechanics*, 1989, doi: 10.1007/BF02321405.
- [5] M. Sutton, C. Mingqi, W. Peters, Y. Chao, and S. McNeill, “Application of an optimized digital correlation method to planar deformation analysis,” *Image and Vision Computing*, vol. 4, no. 3, pp. 143–150, 1986, doi: 10.1016/0262-8856(86)90057-0.
- [6] Y. Su, Q. Zhang, X. Xu, Z. Gao, and S. Wu, “Interpolation bias for the inverse compositional Gauss–Newton algorithm in digital image correlation,” *Optics and Lasers in Engineering*, 2018, doi: 10.1016/j.optlaseng.2017.09.013.
- [7] B. Pan, H. Xie, and Z. Wang, “Equivalence of digital image correlation criteria for pattern matching,” *Applied Optics*, vol. 49, no. 28, p. 5501, Oct. 2010, doi: 10.1364/AO.49.005501.
- [8] Y. L. Dong and B. Pan, “A Review of Speckle Pattern Fabrication and Assessment for Digital Image Correlation,” *Experimental Mechanics*, vol. 57, no. 8, pp. 1161–1181, Oct. 2017, doi: 10.1007/s11340-017-0283-1.
- [9] P. L. Reu, W. Sweatt, T. Miller, and D. Fleming, “Camera System Resolution and its Influence on Digital Image Correlation,” pp. 9–25, 2015, doi: 10.1007/s11340-014-9886-y.
- [10] S. Markidis, S. W. der Chien, E. Laure, I. B. Peng, and J. S. Vetter, “NVIDIA tensor core programmability, performance & precision,” *Proceedings - 2018 IEEE 32nd International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2018*, pp. 522–531, 2018, doi: 10.1109/IPDPSW.2018.00091.
- [11] B. Fitzgerald, “MICVI vTl1 \ f & Opinions Issues The Transformation of Open Source Software1,” *MIS Quarterly*, vol. 30, no. 3, pp. 587–598, 2006, [Online]. Available: <http://www.jstor.org/stable/25148740>

- [12] S. Yoneyama, "Basic principle of digital image correlation for in-plane displacement and strain measurement," *Advanced Composite Materials*, vol. 25, no. 2, pp. 105–123, Mar. 2016, doi: 10.1080/09243046.2015.1129681.
- [13] B. Pan, "Digital image correlation for surface deformation measurement: Historical developments, recent advances and future goals," *Measurement Science and Technology*, vol. 29, no. 8, p. 82001, 2018. doi: 10.1088/1361-6501/aac55b.
- [14] B. Pan, "Digital image correlation for surface deformation measurement: Historical developments, recent advances and future goals," *Measurement Science and Technology*, vol. 29, no. 8, p. 82001, 2018. doi: 10.1088/1361-6501/aac55b.
- [15] N. McCormick and J. Lord, "Digital Image Correlation," *Materials Today*, vol. 13, no. 12, pp. 52–54, Dec. 2010, doi: 10.1016/S1369-7021(10)70235-2.
- [16] B. Pan and K. Li, "A fast digital image correlation method for deformation measurement," *Optics and Lasers in Engineering*, vol. 49, no. 7, pp. 841–847, Jul. 2011, doi: 10.1016/J.OPTLASENG.2011.02.023.
- [17] H. W. Schreier, J. R. Braasch, and M. A. Sutton, "Systematic errors in digital image correlation caused by intensity interpolation," 2000. Accessed: Sep. 06, 2020. [Online]. Available: <http://spiedl.org/terms>
- [18] X.-Y. Liu *et al.*, "Performance of iterative gradient-based algorithms with different intensity change models in digital image correlation," *Optics & Laser Technology*, vol. 44, no. 4, pp. 1060–1067, Jun. 2012, doi: 10.1016/J.OPTLASTEC.2011.10.009.
- [19] P. Bing, Q. Kemaο, X. Huimin, and A. Anand, "Two-dimensional digital image correlation for in-plane displacement and strain measurement: a review," *Measurement Science and Technology*, vol. 20, no. 6, p. 62001, 2009, [Online]. Available: <http://stacks.iop.org/0957-0233/20/i=6/a=062001>
- [20] G. Vend Roux and W. G. Knauss, "Submicron deformation field measurements: Part 2. Improved digital image correlation," *Experimental Mechanics*, vol. 38, no. 2, pp. 86–92, 1998, doi: 10.1007/BF02321649.
- [21] W. Huaiwen and Y. Kang, "Improved digital speckle correlation method and its application in fracture analysis of metallic foil," *Optical Engineering - OPT ENG*, vol. 41, Nov. 2002, doi: 10.1117/1.1511749.
- [22] B. Pan, K. Li, and W. Tong, "Fast, Robust and Accurate Digital Image Correlation Calculation Without Redundant Computations," *Experimental Mechanics*, vol. 53, no. 7, pp. 1277–1289, 2013, doi: 10.1007/s11340-013-9717-6.
- [23] W. H. Peters and W. F. Ranson, "Digital Imaging Techniques In Experimental Stress Analysis," *Optical Engineering*, vol. 21, no. 3, p. 213427, 1982, doi: 10.1117/12.7972925.

- [24] P. F. Luo, Y. J. Chao, M. A. Sutton, and W. H. Peters, “Accurate measurement of three-dimensional deformations in deformable and rigid bodies using computer vision,” *Experimental Mechanics*, vol. 33, no. 2, pp. 123–132, Jun. 1993, doi: 10.1007/BF02322488.
- [25] D. J. Chen, F. P. Chiang, Y. S. Tan, and H. S. Don, “Digital speckle-displacement measurement using a complex spectrum method,” *Applied Optics*, vol. 32, no. 11, p. 1839, Apr. 1993, doi: 10.1364/ao.32.001839.
- [26] B. K. Bay, T. S. Smith, D. P. Fyhrie, and M. Saad, “Digital Volume Correlation: Three-dimensional Strain Mapping Using X-ray Tomography.”
- [27] S. Baker and I. Matthews, “Equivalence and efficiency of image alignment algorithms,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I-1090–I-1097. doi: 10.1109/CVPR.2001.990652.
- [28] W. Tong, “An evaluation of digital image correlation criteria for strain mapping applications,” *Strain*, vol. 41, no. 4, pp. 167–175, Nov. 2005, doi: 10.1111/j.1475-1305.2005.00227.x.
- [29] B. Pan, D. Wu, and L. Yu, “Optimization of a three-dimensional digital image correlation system for deformation measurements in extreme environments,” *Applied Optics*, vol. 51, no. 19, pp. 4409–4419, 2012, doi: 10.1364/AO.51.004409.
- [30] H. Nguyen, Z. Wang, P. Jones, and B. Zhao, “3D shape, deformation, and vibration measurements using infrared Kinect sensors and digital image correlation,” *Applied Optics*, vol. 56, no. 32, p. 9030, Nov. 2017, doi: 10.1364/ao.56.009030.
- [31] P. Bing, Y. LiPing, and Z. QianBing, “SCIENCE CHINA • Review • Review of single-camera stereo-digital image correlation techniques for full-field 3D shape and deformation measurement,” 2018, doi: 10.1007/s11431-017-9090-x.
- [32] L. Yu and B. Pan, “Single-camera stereo-digital image correlation with a four-mirror adapter: optimized design and validation,” *Optics and Lasers in Engineering*, vol. 87, 2016, doi: 10.1016/j.optlaseng.2016.03.014.
- [33] K. Genovese, L. Casaletto, J. A. Rayas, V. Flores, and A. Martinez, “Stereo-Digital Image Correlation (DIC) measurements with a single camera using a biprism,” *Optics and Lasers in Engineering*, vol. 51, no. 3, pp. 278–285, Mar. 2013, doi: 10.1016/j.optlaseng.2012.10.001.
- [34] M. Vo, Z. Wang, L. Luu, and J. Ma, “Advanced geometric camera calibration for machine vision”, doi: 10.1117/1.3647521.
- [35] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000, doi: 10.1109/34.888718.

- [36] W. Burger, “Zhang’s Camera Calibration Algorithm: In-Depth Tutorial and Implementation,” *Technical Report*, 2016, doi: 10.13140/RG.2.1.1166.1688.
- [37] J. Salvi, X. Armangué, and J. Batlle, “A comparative review of camera calibrating methods with accuracy evaluation,” *Pattern Recognition*, vol. 35, no. 7, pp. 1617–1635, Jul. 2002, doi: 10.1016/S0031-3203(01)00126-1.
- [38] J. J. Orteu, “3-D computer vision in experimental mechanics,” *Optics and Lasers in Engineering*, vol. 47, no. 3–4, pp. 282–291, Mar. 2009, doi: 10.1016/j.optlaseng.2007.11.009.
- [39] R. Hartley and A. Zisserman, “Epipolar Geometry and the Fundamental Matrix,” *Multiple View Geometry in Computer Vision*, pp. 239–261, 2011, doi: 10.1017/cbo9780511811685.014.
- [40] “Epipolar geometry - Wikipedia.” https://en.wikipedia.org/wiki/Epipolar_geometry (accessed Sep. 20, 2020).
- [41] “Epipolar geometry - Wikipedia.” https://en.wikipedia.org/wiki/Epipolar_geometry (accessed Sep. 20, 2020).
- [42] B. Chen, H. Liu, B. Pan, and B. Pan, “Calibrating stereo-digital image correlation system using synthetic speckle-pattern calibration target,” *Measurement Science and Technology*, vol. 31, no. 9, Sep. 2020, doi: 10.1088/1361-6501/ab8dfb.
- [43] N. I. Thiruselvam and S. J. Subramanian, “Feature-assisted stereo correlation,” *Strain*, vol. 55, no. 5, Oct. 2019, doi: 10.1111/str.12315.
- [44] R. I. Hartley and P. Sturm, “Triangulation,” 1997.
- [45] P. Reu, “All about speckles: Speckle size measurement,” *Experimental Techniques*, vol. 38, no. 6, pp. 1–2, Nov. 2014, doi: 10.1111/ext.12110.
- [46] P. Reu, “All about speckles: Speckle density,” *Experimental Techniques*, vol. 39, no. 3, pp. 1–2, May 2015, doi: 10.1111/ext.12161.
- [47] P. Reu, “All about speckles: Aliasing,” *Experimental Techniques*, vol. 38, no. 5, pp. 1–3, Sep. 2014, doi: 10.1111/ext.12111.
- [48] P. Reu, “All about speckles: Edge sharpness,” *Experimental Techniques*, vol. 39, no. 2, pp. 1–2, Mar. 2015, doi: 10.1111/ext.12139.
- [49] G. Crammond, S. W. Boyd, and J. M. Dulieu-Barton, “Speckle pattern quality assessment for digital image correlation,” *Optics and Lasers in Engineering*, vol. 51, no. 12, pp. 1368–1378, Dec. 2013, doi: 10.1016/J.OPTLASENG.2013.03.014.
- [50] S. Yoneyama, A. Kitagawa, S. Iwata, K. Tani, and H. Kikuta, “BRIDGE DEFLECTION MEASUREMENT USING DIGITAL IMAGE CORRELATION,” *Experimental*

- Techniques*, vol. 31, no. 1, pp. 34–40, Jan. 2007, doi: 10.1111/j.1747-1567.2006.00132.x.
- [51] B. Pan, D. Wu, and Y. Xia, “An active imaging digital image correlation method for deformation measurement insensitive to ambient light,” *Optics & Laser Technology*, vol. 44, no. 1, pp. 204–209, Feb. 2012, doi: 10.1016/J.OPTLASTEC.2011.06.019.
- [52] S. Vanlanduit, J. Vanherzeele, R. Longo, and P. Guillaume, “A digital image correlation method for fatigue test experiments,” *Optics and Lasers in Engineering*, vol. 47, no. 3–4, pp. 371–378, Mar. 2009, doi: 10.1016/j.optlaseng.2008.03.016.
- [53] K. Genovese, L. Casaletto, J. A. Rayas, V. Flores, and A. Martinez, “Stereo-Digital Image Correlation (DIC) measurements with a single camera using a biprism,” *Optics and Lasers in Engineering*, vol. 51, no. 3, pp. 278–285, Mar. 2013, doi: 10.1016/j.optlaseng.2012.10.001.
- [54] K. Genovese, L. Casaletto, J. A. Rayas, V. Flores, and A. Martinez, “Stereo-Digital Image Correlation (DIC) measurements with a single camera using a biprism,” *Optics and Lasers in Engineering*, vol. 51, no. 3, pp. 278–285, Mar. 2013, doi: 10.1016/j.optlaseng.2012.10.001.
- [55] S. Baker and I. Matthews, “Lucas-Kanade 20 years on: A unifying framework,” *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, Feb. 2004, doi: 10.1023/B:VISI.0000011205.11775.fd.
- [56] X. Shao, X. Dai, and X. He, “Noise robustness and parallel computation of the inverse compositional Gauss-Newton algorithm in digital image correlation,” *Optics and Lasers in Engineering*, 2015, doi: 10.1016/j.optlaseng.2015.03.005.
- [57] Z. Jiang, Q. Kemao, H. Miao, J. Yang, and L. Tang, “Path-independent digital image correlation with high accuracy, speed and robustness,” *Optics and Lasers in Engineering*, vol. 65, pp. 93–102, Feb. 2015, doi: 10.1016/j.optlaseng.2014.06.011.
- [58] L. Luu, Z. Wang, M. Vo, T. Hoang, and J. Ma, “Accuracy enhancement of digital image correlation with B-spline interpolation,” *Optics Letters*, vol. 36, no. 16, p. 3070, 2011, doi: 10.1364/ol.36.003070.
- [59] X. Dai, X. He, X. Shao, and Z. Chen, “Real-time 3D digital image correlation method and its application in human pulse monitoring,” *Applied Optics*, Vol. 55, Issue 4, pp. 696–704, vol. 55, no. 4, pp. 696–704, Feb. 2016, doi: 10.1364/AO.55.000696.
- [60] Y. Xue *et al.*, “High-accuracy and real-time 3D positioning, tracking system for medical imaging applications based on 3D digital image correlation,” *Optics and Lasers in Engineering*, vol. 88, pp. 82–90, Jan. 2017, doi: 10.1016/J.OPTLASENG.2016.07.002.
- [61] B. Pan, L. Tian, and X. Song, “Real-time, non-contact and targetless measurement of vertical deflection of bridges using off-axis digital image correlation,” *NDT & E International*, vol. 79, pp. 73–80, Apr. 2016, doi: 10.1016/J.NDTEINT.2015.12.006.

- [62] R. Wu, C. Kong, & K. Li, and & D. Zhang, “Real-Time Digital Image Correlation for Dynamic Strain Measurement,” *Experimental Mechanics*, doi: 10.1007/s11340-016-0133-6.
- [63] M. L. H. Y. G. Z. Z. C. C Tang, “The improved genetic algorithms for digital image correlation method,” *Chin Opt Lett*, vol. 2, no. 10, pp. 574–577, 2004.
- [64] R. Storn and K. Price, “Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces,” *J Glob Optim*, vol. 11, no. 4, pp. 341–359, 1997, doi: 10.1023/a:1008202821328.
- [65] J. Kennedy and R. Eberhart, “Particle swarm optimization,” *Proceedings of ICNN’95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.
- [66] “Single Board Computers Explained: Which is Right for Me?” <https://www.circuito.io/blog/single-board-computer/> (accessed Oct. 10, 2020).
- [67] “Teach, learn, and make with the Raspberry Pi Foundation.” <https://www.raspberrypi.org/> (accessed Jul. 26, 2022).
- [68] “BeagleBoard.org - bone.” <https://beagleboard.org/bone> (accessed Jul. 26, 2022).
- [69] “Jetson AGX Xavier Developer Kit | NVIDIA Developer.” <https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit> (accessed Jul. 26, 2022).
- [70] “NVIDIA Jetson Nano Developer Kit | NVIDIA Developer.” <https://developer.nvidia.com/embedded/jetson-nano-developer-kit> (accessed Jul. 26, 2022).
- [71] “Tinker Board.” <https://tinker-board.asus.com/product/tinker-board.html> (accessed Jul. 26, 2022).
- [72] G. Halfacree, “Raspberry Pi 4 now comes with 2GB RAM Minimum,” *The MagPi*, no. 91, 2020, Accessed: Oct. 10, 2020. [Online]. Available: <https://magpi.raspberrypi.org/issues/91/pdf>
- [73] L. Zhang *et al.*, “High accuracy digital image correlation powered by GPU-based parallel computing,” *Optics and Lasers in Engineering*, vol. 69, pp. 7–12, 2015, doi: 10.1016/j.optlaseng.2015.01.012.
- [74] M. Gates, M. T. Heath, and J. Lambros, “High-performance hybrid CPU and GPU parallel algorithm for digital volume correlation,” *The International Journal of High Performance Computing Applications*, vol. 29, no. 1, pp. 92–106, Feb. 2015, doi: 10.1177/1094342013518807.

- [75] A. Singh and S. N. Omkar, "Digital Image Correlation using GPU Computing Applied to Biomechanics," *Biomedical Science and Engineering*, vol. 1, no. 1, pp. 1–10, 2013, doi: 10.12691/BSE-1-1-1.
- [76] T. Wang, Z. Jiang, Q. Kema, F. Lin, and S. H. Soon, "GPU Accelerated Digital Volume Correlation," *Experimental Mechanics*, vol. 56, no. 2, pp. 297–309, Feb. 2016, doi: 10.1007/s11340-015-0091-4.
- [77] A. Deshpande and D. Riehle, "The total growth of open source," *IFIP International Federation for Information Processing*, vol. 275, no. December 2006, pp. 197–209, 2008, doi: 10.1007/978-0-387-09684-1_16.
- [78] D. Z. Turner, "Digital Image Correlation Engine (DICE) Reference Manual," *Sandia Nat. Lab*, no. Sandia Rep. SAND2015-10606 O, 2015, [Online]. Available: <http://dicengine.github.io/dice/>
- [79] "Linking Error on DICE_TrackingMovieMaker · Issue #136 · dicengine/dice." <https://github.com/dicengine/dice/issues/136> (accessed Dec. 05, 2020).
- [80] "Gstreamer Features." <https://gstreamer.freedesktop.org/features/> (accessed Nov. 28, 2021).
- [81] "GStreamer Basics." <https://gstreamer.freedesktop.org/documentation/tutorials/basic/concepts.html?gi-language=c> (accessed Nov. 28, 2021).
- [82] A. M. Curry, "Design, manufacture and commissioning of a low pressure quasistatic bulge tester for skin and membrane tissue." Accessed: Feb. 06, 2022. [Online]. Available: <https://open.uct.ac.za/handle/11427/32689>
- [83] "Charles Jekel - jekel.me - Least Squares Sphere Fit." <https://jekel.me/2015/Least-Squares-Sphere-Fit/> (accessed Feb. 06, 2022).
- [84] A. Singh and S. N. Omkar, "Digital Image Correlation using GPU Computing Applied to Biomechanics," *Biomedical Science and Engineering*, vol. 1, no. 1, pp. 1–10, 2013, doi: 10.12691/BSE-1-1-1.
- [85] Y. Su, Q. Zhang, and Z. Gao, "Statistical model for speckle pattern optimization," *Optics Express*, vol. 25, no. 24, p. 30259, Nov. 2017, doi: 10.1364/oe.25.030259.

7. Appendices

7.1. Appendix 1

	Deformed Image	Moving Window									$\sum F(x,y)G(x^*,y^*)$									
												79840	109175	83682	128331	76459	90863	78595	65687	65087
0	23	217									4991	0	0	0	0	0	0	0	0	0
1	146	89	217								12994	31682	0	0	0	0	0	0	0	0
2	57	170	89	217							9690	5073	12369	0	0	0	0	0	0	0
3	217	64	170	89	217						13888	36890	19313	47089	0	0	0	0	0	0
4	89	113	64	170	89	217					10057	5696	15130	7921	19313	0	0	0	0	0
5	170	166	113	64	170	89	217				28220	19210	10880	28900	15130	36890	0	0	0	0
6	64		166	113	64	170	89	217			0	10624	7232	4096	10880	5696	13888	0	0	
7	113			166	113	64	170	89	217		0	0	18758	12769	7232	19210	10057	24521	0	
8	166				166	113	64	170	89	217	0	0	0	27556	18758	10624	28220	14774	36022	
9	31					166	113	64	170	89	0	0	0	0	5146	3503	1984	5270	2759	
10	90						166	113	64	170	0	0	0	0	0	14940	10170	5760	15300	
11	86							166	113	64	0	0	0	0	0	0	14276	9718	5504	
12	34								166	113	0	0	0	0	0	0	0	5644	3842	
13	10									166	0	0	0	0	0	0	0	0	1660	
											$\frac{\sum F(x,y)G(x^*,y^*)}{\sqrt{\sum F(x,y)^2 \sum G(x^*,y^*)^2}}$									
											0.138867	0.18989	0.14555	0.223208	0.132987	0.15804	0.171453	0.17328	0.194783	

7.2. Appendix 2

		Subsets X								Average Error
		1	2	3	4	5	6	7	8	
Subsets Y	1	0.97438	1.0035	0.96472	1.0034	0.98407	0.97446	0.9898	0.99226	Target: 1mm
	2	0.98161	0.97006	0.97247	0.97295	0.96438	0.98941	0.96163	0.9795	
	3	0.96425	0.98177	0.96715	0.97115	0.96964	0.94984	0.97497	0.99294	
	4	0.94249	0.97094	0.9677	0.96252	0.98178	0.97816	0.97128	0.9686	
	5	0.9558	0.96964	0.9527	0.99609	0.98348	0.97839	0.99578	0.98914	
	6	0.97416	0.97175	0.97899	0.99343	0.994	0.98131	0.96771	0.99047	
	7	0.97795	0.99079	0.98533	0.97045	1.0073	0.9677	0.96015	0.97818	
	8	0.96888	0.99268	0.98744	0.97014	0.97671	0.96151	0.96386	1.0063	
	9	0.97397	0.98329	0.96895	0.986	0.9852	0.96418	0.98179	0.99424	
	10	0.98202	0.99749	0.97821	0.99204	0.98792	0.98528	0.97753	0.98023	
	11	0.95777	0.97638	0.9743	0.9788	0.98062	0.98175	0.98583	0.98674	
	12	0.98543	0.97592	0.98581	0.97614	0.98283	0.97828	0.9801	0.96455	
	13	0.96618	0.99184	0.99462	0.98198	0.97492	0.9871	0.99094	0.97044	
	14	0.95894	0.9723	0.97706	0.97355	0.97395	0.97865	0.97592	0.97709	
	15	0.98839	0.99739	0.98065	1.0047	0.96796	0.99412	0.98156	0.98449	
		1	2	3	4	5	6	7	8	
Subsets Y	1	2.0195	2.0328	1.9988	2.0178	2.007	2.0047	1.9888	2.0076	Target: 2mm
	2	1.994	1.9979	2.0088	2.0182	1.9919	2.0195	1.9565	1.9976	
	3	1.9765	1.9996	1.9913	1.9989	1.9927	2.013	1.9957	2.007	
	4	1.9595	1.975	1.9775	1.9813	2.0099	1.9936	2.0005	1.9994	
	5	1.9845	1.9816	1.9687	1.988	1.9978	1.9942	2.017	2.0112	
	6	2.0245	1.9988	2.0055	2.0223	2.0136	1.9837	2.0037	2.0024	
	7	2.0126	2.0147	1.9981	1.9876	2.0109	1.9834	1.9711	1.991	
	8	1.9694	1.9969	2.0149	1.9801	2.001	1.9869	2.0096	2.0209	
	9	2.0014	1.9921	1.965	1.9874	2.0028	1.9962	2.0004	2.0566	
	10	2.0133	2.0367	1.985	1.9827	1.9852	2.0179	2.0082	2.002	
	11	1.9994	2.0007	2.0106	2.0176	1.9896	1.9854	1.99	2.0073	
	12	1.9914	2.0031	1.9869	2.0022	1.9975	2.0031	1.9801	1.9818	
	13	1.9668	1.9956	2.0101	1.994	1.9857	1.9865	2.0012	2.0146	
	14	1.9807	2.0108	1.9765	1.9633	2	2.0073	1.9974	2.0019	
	15	2.0268	2.0188	1.9965	2.0156	1.9732	2.0182	2.0017	2.0046	
		1	2	3	4	5	6	7	8	
Subsets Y	1	2.9096	2.9578	2.916	2.9712	2.9329	2.9428	2.9167	2.9549	Target: 3mm
	2	2.9469	2.9439	2.9036	2.9236	2.938	2.9678	2.8977	2.9305	
	3	2.9015	2.9328	2.9203	2.9297	2.8803	2.9088	2.9214	2.9483	
	4	2.8961	2.9208	2.911	2.9237	2.955	2.9339	2.8981	2.944	
	5	2.8852	2.9332	2.9174	2.9324	2.9314	2.949	2.9591	2.9407	
	6	2.9099	2.9172	2.9316	2.9453	2.9472	2.932	2.9421	2.9491	
	7	2.9597	2.939	2.9141	2.8942	2.9527	2.9303	2.9043	2.9308	
	8	2.924	2.9419	2.979	2.919	2.9079	2.8803	2.946	2.9497	
	9	2.9366	2.9368	2.8964	2.9274	2.9438	2.9309	2.9183	2.9477	
	10	2.9336	2.9631	2.9239	2.9118	2.918	2.9454	2.9646	2.9423	
	11	2.8796	2.927	2.9348	2.9433	2.934	2.934	2.9241	2.9571	
	12	2.9347	2.9401	2.9098	2.9276	2.9199	2.9236	2.9348	2.9223	
	13	2.9031	2.941	2.9482	2.9171	2.9031	2.8982	2.9265	2.9726	
	14	2.9495	2.9605	2.9106	2.9064	2.9485	2.94	2.9043	2.9253	
	15	2.9574	2.9613	2.9449	2.9403	2.9083	2.9528	2.9502	2.9376	
		1	2	3	4	5	6	7	8	
Subsets Y	1	3.944	3.9552	3.9185	3.96	3.9394	3.9359	3.9476	3.9512	Target: 4mm
	2	3.9482	3.9473	3.9299	3.9595	3.9241	3.9589	3.9061	3.9306	
	3	3.9202	3.9388	3.932	3.9136	3.9049	3.9226	3.9153	3.9385	
	4	3.8893	3.9113	3.9381	3.9316	3.9624	3.9151	3.9153	3.9703	
	5	3.8718	3.9328	3.9178	3.9319	3.9445	3.9667	3.9588	3.9281	
	6	3.9233	3.9287	3.9095	3.9585	3.9544	3.9272	3.9538	3.968	
	7	3.9707	3.9104	3.9032	3.9315	3.9297	3.9312	3.9126	3.9208	
	8	3.9228	3.9576	3.9681	3.9242	3.9138	3.8913	3.9305	3.9587	
	9	3.9446	3.9399	3.9307	3.9396	3.9546	3.9208	3.922	3.9538	
	10	3.9156	3.9752	3.9255	3.9001	3.9332	3.9386	3.9736	3.9216	
	11	3.9052	3.9212	3.9123	3.9195	3.9595	3.9283	3.9292	3.9803	
	12	3.9489	3.9264	3.9414	3.9284	3.8983	3.9317	3.9292	3.9213	
	13	3.957	3.9542	3.974	3.9182	3.8988	3.9303	3.9195	3.9481	
	14	3.9436	3.942	3.9206	3.9051	3.9809	3.9326	3.9177	3.9341	
	15	3.9537	3.9534	3.9348	3.9379	3.8999	3.9767	3.9359	3.9326	

		1	2	3	4	5	6	7	8	
Subsets Y	1	2.929	2.9843	2.944	2.9986	2.9592	2.9617	2.9355	2.9711	
	2	2.9608	2.9615	2.9265	2.9484	2.9573	2.9947	2.9101	2.9596	
	3	2.9181	2.9546	2.9381	2.9375	2.9092	2.9416	2.9472	2.9737	
	4	2.9139	2.9381	2.9317	2.9444	2.9701	2.946	2.9306	2.9736	Target:
	5	2.9136	2.9516	2.9402	2.9553	2.9471	2.977	2.9742	2.9541	3mm
	6	2.9361	2.9422	2.9542	2.9754	2.9629	2.9547	2.9632	2.9739	
	7	2.9708	2.9522	2.9403	2.9205	2.9788	2.9596	2.9298	2.9487	
	8	2.9379	2.9574	2.9881	2.9383	2.9319	2.9064	2.9681	2.9662	
	9	2.9594	2.957	2.9163	2.9318	2.9614	2.9386	2.9411	2.971	
	10	2.9597	2.9919	2.9475	2.9245	2.9371	2.9627	2.9789	2.95	
	11	2.9099	2.9555	2.9646	2.968	2.9596	2.9517	2.9468	2.9796	Error:
	12	2.9488	2.9497	2.933	2.9513	2.9442	2.9514	2.9506	2.9386	0.0486
	13	2.9184	2.9641	2.9641	2.9299	2.9216	2.9196	2.9512	2.9913	
	14	2.9698	2.9809	2.9286	2.9241	2.9608	2.9554	2.9308	2.9426	
	15	2.9836	2.9883	2.9637	2.9617	2.926	2.9799	2.9681	2.9584	
		1	2	3	4	5	6	7	8	
Subsets Y	1	2.0449	2.0566	2.0204	2.0448	2.0287	2.0216	2.0086	2.0249	
	2	2.0137	2.0119	2.0303	2.0357	2.0093	2.0398	1.9719	2.0164	
	3	1.9907	2.017	2.0067	2.0083	2.0132	2.0288	2.0188	2.0267	
	4	1.9813	1.9914	1.9897	1.9996	2.0234	2.0136	2.0188	2.0282	Target:
	5	2.0105	1.9991	1.9919	2.0107	2.0132	2.0183	2.0344	2.0295	2mm
	6	2.0428	2.0191	2.0268	2.0414	2.0292	2.0082	2.0286	2.0163	
	7	2.0339	2.0328	2.0157	2.005	2.0282	2.0029	1.9929	2.0108	
	8	1.9938	2.0091	2.0311	1.999	2.0219	2.0039	2.0291	2.0347	
	9	2.017	2.0096	1.982	1.9997	2.0161	2.0113	2.0187	2.0743	
	10	2.0288	2.055	1.9993	1.9993	1.9993	2.0274	2.0259	2.0181	
	11	2.0141	2.02	2.0295	2.0402	2.0121	1.9979	2.0056	2.0289	Error:
	12	2.0121	2.0192	2.0049	2.0205	2.0177	2.0092	2.0017	2.0004	0.0191
	13	1.9843	2.0085	2.0221	2.0066	2	2.0083	2.0184	2.0362	
	14	2.0007	2.0324	1.9915	1.9845	2.0108	2.0145	2.0181	2.0227	
	15	2.047	2.0415	2.0112	2.033	1.9897	2.0314	2.0138	2.0174	
		1	2	3	4	5	6	7	8	
Subsets Y	1	1.054	1.0753	1.0426	1.0872	1.0515	1.0534	1.0627	1.0656	
	2	1.055	1.0419	1.0551	1.0636	1.0455	1.0606	1.0194	1.0503	
	3	1.037	1.0595	1.041	1.0471	1.0428	1.0487	1.047	1.0668	
	4	1.02	1.0419	1.044	1.035	1.048	1.0503	1.0638	1.0416	Target:
	5	1.0392	1.0406	1.0274	1.0668	1.0547	1.0598	1.071	1.069	1mm
	6	1.054	1.0546	1.0499	1.0747	1.0678	1.0552	1.0249	1.0643	
	7	1.0456	1.062	1.0557	1.0514	1.0815	1.0468	1.0329	1.0518	
	8	1.0421	1.0616	1.0573	1.0544	1.0488	1.0504	1.0531	1.0749	
	9	1.0403	1.0576	1.0298	1.0529	1.0572	1.0523	1.0694	1.0799	
	10	1.0527	1.0714	1.039	1.0533	1.051	1.063	1.0459	1.0591	
	11	1.0352	1.0464	1.0452	1.061	1.0469	1.0546	1.0623	1.0624	Error:
	12	1.0445	1.0454	1.0699	1.0535	1.0432	1.0534	1.0453	1.0398	0.0525
	13	1.0357	1.0618	1.0572	1.0592	1.0482	1.0576	1.0647	1.045	
	14	1.0244	1.0533	1.0449	1.0293	1.0444	1.0528	1.0584	1.048	
	15	1.053	1.0725	1.0599	1.0601	1.0439	1.0674	1.0675	1.0554	

7.3. Appendix 3

```
S,05 //Set the sample period
F,01 //Move forward 1mm
W,01 //Wait 1 second
F,01
W,01
F,01
W,01
F,01
W,01
F,01
W,01
F,01
W,01
F,01
W,01
F,01
W,01
F,01
W,01
F,01
W,01
F,01
W,10 //Wait 10 seconds
END //End the script
```

7.4. Appendix 4

The results repository can be cloned to your local machine. If cloned to your local machine, please note that you will need the following installed on your machine to view the analysis files:

- Git – to clone the repositories
 - `git clone https://github.com/haemishkyd/kydhae001_msc_results.git`
- Python 3.6 or newer with the following packages installed:
 - Pandas
 - Matplotlib
 - Numpy
- Jupyter Notebooks

The source code can be cloned to your local machine. You will need the following on your machine to view the code:

- Git – to clone the repositories
 - `git clone https://github.com/haemishkyd/kydhae001_msc_code.git`

Please note: In order to actually build this code to an executable will require significant effort.

- A machine with the correct operating system will need to be obtained.
- The DICE, Trilinos and OpenCV code bases will need to be built.
- The Cmake files will need to be updated to reflect the location of these built code bases.

The explanation of how to build this is not provided in any detail here, but if this is something that is required the author can be contacted.

7.5. Appendix 5

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('ref.tif', 0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))

plt.subplot(121), plt.imshow(img, cmap='gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()

rows, cols = img.shape
crow, ccol = rows/2, cols/2
window_size = 60
fshift[int(crow-(window_size/2)):int(crow+(window_size/2)), int(ccol-(window_size/2)):int(ccol+(window_size/2))] = 0
f_ishift = np.fft.ifftshift(fshift)
img_back = np.fft.ifft2(f_ishift)
img_back = np.abs(img_back)

plt.subplot(121), plt.imshow(img, cmap='gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(img_back, cmap='gray')
plt.title('Image after HPF'), plt.xticks([]), plt.yticks([])

plt.show()
```