An investigation of the strain field and inter-fragmentary movement of a broken hemi-pelvis

By

Hamed Nazeri

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering
University of Alberta

Abstract

Pelvic fracture is considered a common and sometimes fatal injury. There are still developments in novel pelvic fixations. In order to investigate the performance of these fixations, a specialized testing set-up is required. Typically, the strains of a few selected points and the changes in the boundary condition on a pelvis are evaluated. The goal of the current study was to investigate the strain field on a region of interest of a broken hemi-pelvis as well as the inter-fragmentary movement in 3D. Both goals have not been achieved in other studies yet.

A multi-axis biomechanical testing apparatus (MABTA) was previously developed in the Department of Mechanical Engineering of the University of Alberta. The current study investigates modifications incorporated into the MABTA such as the force direction exerted by the prosthetic femur. Novel criteria for determining the loading direction is presented. Also the errors associated with the apparatus were evaluated where the error of the motor and load cell were determined.

In order to investigate the strain field in a selected region of interest, the MABTA was equipped with a 3D digital image correlation (DIC) system. The error of pure movement using the DIC system was determined at 3.5%, and the error of measuring the strain field was less than 6%. The novel aspect of the work is to use DIC for measuring the gap opening in three directions. It allows investigating the value and mode of opening anywhere in the region of interest. In order to measure the gap opening, two Matlab programs were developed and tested to have the accuracy of 5 µm for in-plane opening and 30 µm for out-of-plane opening. Finally, two tests on an intact and a broken hemi-pelvis were done, and the results of the strain field and the inter-fragmentary movement were obtained.

“Scientia potentia est”

Knowledge is power

Attributed to Sir Francis Bacon [1561–1617]

This thesis is dedicated to my beloved parents, Mahmoud and Ramesh,

And to my dearest grandfather, Reza, for his unconditional love, kindness, and supports

Acknowledgments

Besides my supervisor Dr. Kajsa Duke, I would like to thank my co-supervisor Dr. David Nobes for his guidance and Dr. Yongsheng Ma for his financial support. This work was not possible without the kind help of Ms. Roya Vaghar, Mr. Bernie Faulkner, and Ms. Fatemeh Khademi.

I would also like to appreciate the University of Alberta which made it possible for me to pursue my graduate studies.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

I. Motivation

Among different injuries that may occur during daily routine, pelvic injuries may be fatal and are often difficult to treat. They are difficult simply because the pelvis forms the skeleton supporting the upper body, and undergoes huge amount of force which is often several times body weight [1]. Moreover, due to the fact that some arteries pass through the pelvic ring such as the superior gluteal artery, fatal arterial injuries could be found in those with pelvis fractures [2], [3]. Diagnosed with a pelvic fracture, the broken bone needs to be immobilized in order to heal. In many cases, casting could keep the parts together, whereas in this type of fracture using a cast is not feasible. Therefore, some specialized fixations have been developed.

The necessity for developing pelvic fracture treatments requires evaluation of their performance. This is usually addressed by testing the fixation at *in-vitro* conditions using a testing apparatus. Several testing apparatuses have been developed as will be discussed in section II. However, this evaluation is often limited to determining the strain of a finite number of points. Also, if the crack opening is investigated, the points of interest were determined beforehand to install the measuring device.

The motivation of the current work is to investigate the strain field on the surface of a hemi-pelvis as well as measuring the inter-fragmentary movements. The proposed technique to achieve the goals is digital image correlation (DIC) which will be explained in section III. Also the loading and the displacement of the femur, by which the pelvis is usually loaded, is of interest.

This study is among the few determining the strain field on the pelvis, and is the only one which determines the strain field on the broken hemi-pelvis. The novel aspect of the work is to use DIC to investigate the crack opening in three directions. The advantage of using the DIC for this purpose is that after implementing the DIC method in the area of interest, it can measure the gap opening anywhere on the tested area. This creates an opportunity to investigate the mode of opening in different location of the crack. Having a successful test on a broken hemi-pelvis, the goal would be achieved.

II. Literature Review

i. Human Anatomy and Pelvis Bone

Many references [4]–[6] define the anatomical planes as illustrated in Figure 1. There are three anatomical planes, namely, the sagittal, coronal (frontal), and transverse planes. The sagittal plane is the vertical plane that passes through the long axis of the body dividing it into left and right halves and bisects the posterior and anterior planes. Transverse plane passes through the body at right angles to the body long axis and the sagittal plane dividing the body into superior and inferior sections. The coronal or frontal plane is a vertical plane that passes through the body at right angles to the sagittal and transverse planes dividing the body into anterior and posterior sections. The method for defining a standardized coordinate system suggests the Z and Y axes are toward superior and anterior respectively, and the X axis is obtained from the right hand rule.

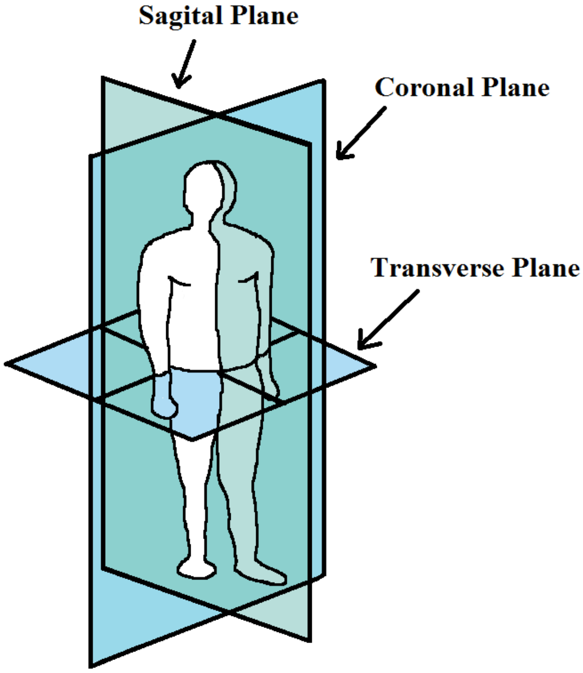


Figure 1: The anatomical planes of human body

The pelvis is a ring-like bone structure connecting the spine to the lower trunk. Each hemi-pelvis is made of three bones: ilium, ischium, and pubis. These bones are initially separated and by aging they grow together. Strong ligaments connect the hemi-pelvises to the sacrum. The sacrum is a triangular bone attached to the base of the spine. A hollow cup named acetabulum serves as the socket for the hip joint on each hemi-pelvis [7]. The pelvis is sometimes subjected to forces that may be up to six times body weight [8]. When loaded, the thin layer of the cortical bone resists to deform and the trabecular bone within have negligible effect [9]. The loads are transferred from the femur to the upper trunk by the acetabulum; therefore, the acetabulum is an important weight bearing structure of the pelvis.

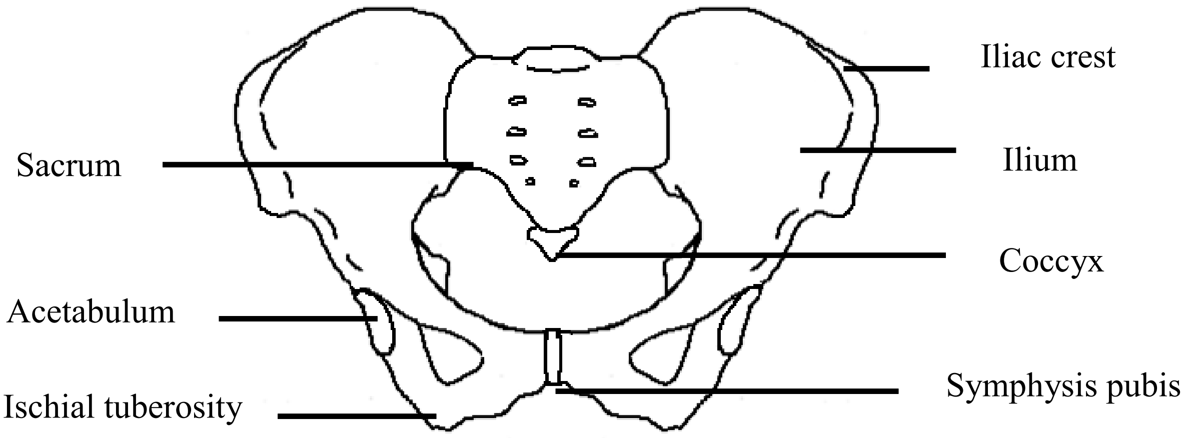


Figure 2: The female pelvis

Finding the anatomical planes for the pelvis has been studied before [10], [11]. The first step is to find the landmarks on the pelvis as follows. These landmarks are illustrated in Figure 3:

- RASIS: right anterior superior iliac spine.
- LASIS: left anterior superior iliac spine.
- RPSIS: right posterior superior iliac spine.
- LPSIS: left posterior superior iliac spine.

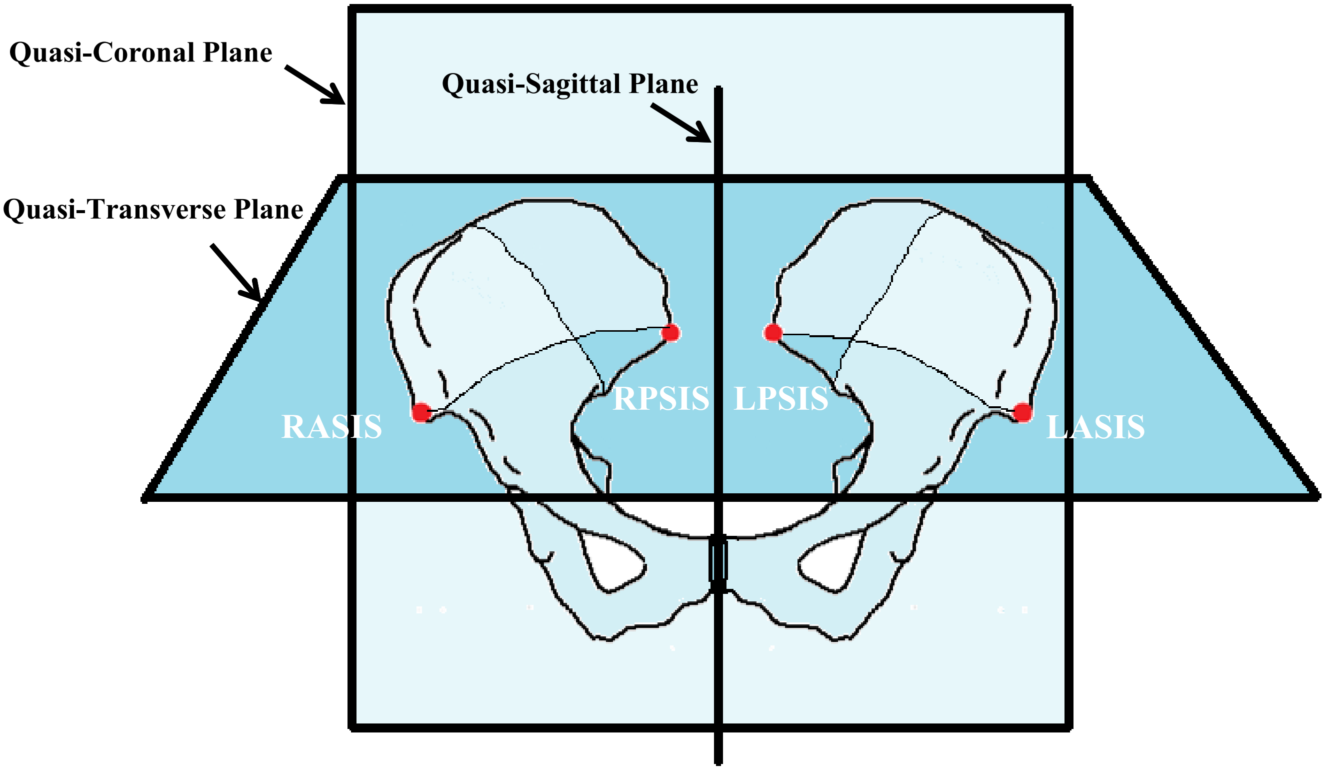


Figure 3: Anatomical landmarks on the pelvis

Considering that each individual has its unique body which is not perfectly symmetrical, just an approximation of the anatomical planes can be found. These estimated planes are specified by the term: “quasi”. In the pelvis, the landmarks give an estimation of the planes as well. Using the LASIS, RASIS, and the midpoint between RPSIS and LPSIS, the quasi-transverse plane can be found. At a right angle to this plane and containing LASIS and RASIS points, the quasi-coronal plane can be found, and finally the quasi-sagittal plane would be orthogonal to the both previous planes passing through the midpoint of either PSIS or ASIS.

ii. Pelvis Fracture and Fixations

Pelvis fracture may occur from both repetitive and sudden loadings. The former is categorized into two different subcategories: where there is abnormal loading to a normal bone, also known as "fatigue fractures" or there is the normal loading to bone weakened by demineralization, "insufficiency fractures" [12], [13]. Insufficiency fracture is commonly seen in the elderly who suffer from osteoporosis or metastatic diseases [14].

The other type of fracture (due to sudden loadings) is classified by two well-known methods. The Tile method, developed in 1980 [15] and the Young-Burgess method, developed in 1990 [16]. A brief classification proposed by Tile is as follows:

1. Type A: when the pelvis is still stable
2. Type B: when the pelvis unstable rotationally, but is vertically stable
3. Type C: when the pelvis is unstable both rotationally and vertically

Young-Burgess method classifies the pelvic fractures by the direction of the force resulting in the fracture as follows:

1. APC: Anterior-posterior compression
2. LC: Lateral compression
3. VS: vertical shear

The main reason of pelvic fractures is sudden high-energy forces. One major source is those forces generated by motor vehicle such as collisions, and crush accident or the other source is falling. Depending on the direction and degree of the force, these fractures may lead to arterial and neural injuries that might be life-threatening and require immediate surgical treatment [7].

In the United States, an estimated of 44 million Americans or approximately 55 percent of people 50 years of age and older are at risk of osteoporosis. Approximately 10 million of them already have this disease and the rest 34 million have low bone mass which may lead to the onset of this disease. Elderly people with osteoporosis are subjected to the risk of pelvic fracture even during a fall from standing. This could happen when getting out of the bathtub or descending stairs too [7].

It is estimated that one in every two women and one in every four men older than 50 will have an osteoporosis-related fracture in their lifetime. In 2005, almost 2 million fractures were seen due to osteoporosis. It is predicted that by 2025 almost 3 million general fractures will be seen annually because of this disease [17].

This disease is more commonly seen in Canada. Around 1.5 million Canadians older than 40 (approximately 10% of those over 40), reported that they were suffering from osteoporosis, in that group, women were 4 times more likely have osteoporosis than men [18]. At least 1 in 3 women and 1 in 5 men will experience an osteoporosis related fracture during their lifetime [19].

Generally, 10% of the injuries to the human skeleton are those related to pelvic ring, 68% of which occurred when having an accident with a motor vehicle [20].

In 2010, the number of collisions in Canada was around 124,000 [21]. Typically 58.2% of collisions are frontal, and 5.5% of frontal collisions result in pelvic injuries [22]. Also, 6.4% of collisions are left lateral of which 20% result in pelvic fracture. So it can be estimated that annually more than 5,500 pelvic fractures happen in Canada due to car accidents.

Inter-fragmentary screws and locking plates are the more frequently used types of pelvic fixations [23], [24]. The fixations in which memory alloys are employed, are another novel developing fixations [25]. Possible flaws in inserting the lag screws following by emerging strain concentration in the pelvis may lead to re-fracture of the pelvis [26]. New efforts are made to improve more reliable fixation.

In order to develop the pelvis fixations both experimental and numerical methods have been employed. Various commercial FE software (including Abaqus, Ansys, etc.) were implemented to investigate their performance. FE models may offer biomechanical planning for stabilizing fractures, which has been an ongoing effort in the area of biomechanics. This type of numerical simulations, in order to reach higher stability of osteosyntheses of the pelvis, has been implemented by only a few groups working in biomechanics area. Finite element method is used for predicting the inter-fragmentary movements and finding the optimum screw position [24] as well as finding the maximum strain and dislocation in models [27]. There can be found other studies using numerical analysis as well [28], [29].

iii. Testing Apparatuses

For the purpose of testing a pelvis response to loading applied vertically or in some cases laterally, many testing setups have been developed. As the measured resulting variable, a number of different parameters may be sought. In some experiments, only the response to the boundary conditions (B.C.s) such as femur loading and displacement is measured [30]–[34] whereas in some others, the strain in a few points of interest is looked at [35], [36], and in some studies only the displacement of the broken part has been investigated [24], [37]. Although the strain field on the pelvis surface has just recently been investigated [38], [39], the inter-fragmentary movement was not looked into at the same time. The multi-axis biomechanical testing apparatus (MABTA) was also developed in 2006 in the Mechanical Engineering Department at the University of Alberta [71]. It was originally built for testing on the knee, but it is capable of being used on similar testing on the pelvis bone [71].

Different methods of holding the pelvis have been implemented in experimental testing. Usually the sacrum is kept fixed and the force is exerted by a prosthetic or an actual femur head on the acetabulum cup (the latter is commonly used when a cadaveric pelvis is tested). However, in some cases the acetabulum is fixed and the loading is done by the sacrum [30]. Of course, if the lateral loading (for modeling an impact situation) is used [34], [40], the iliac crest may be restricted on one side of the pelvis.

For femoral loading angle, even though many different setups have been developed, in most of them a vertical force is exerted at the pelvis corresponding to standing position [29], [30], [35], [41]. In very few studies the loading angle represents one-leg standing [33], [42].

The tests are done either on a complete pelvis (two hip bones and a sacrum) [29], [36] or on a hemi-pelvis (just a hip bone) [23], [35], [38]. In case of using a hemi-pelvis, in some studies the pubic joint has been left without any support [35] as can be seen in Figure 4, or restrained as a part of the boundary condition[38]. Figure 5 shows a recently developed testing setup in which the pubic symphysis is potted too.

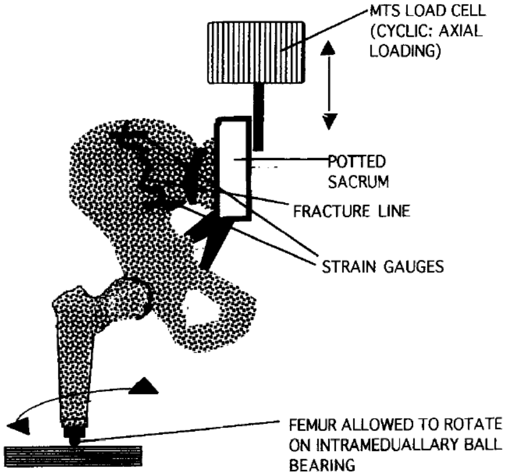

MTS LOAD CELL (CYCLIC: AXIAL LOADING)
POTTED SACRUM
FRACTURE LINE
STRAIN GAUGES
FEMUR ALLOWED TO ROTATE ON INTRAMEDUALLARY BALL BEARING

Figure 4: Previously developed setup for testing a hemi-pelvis where the pubic joint is not restrained[1]

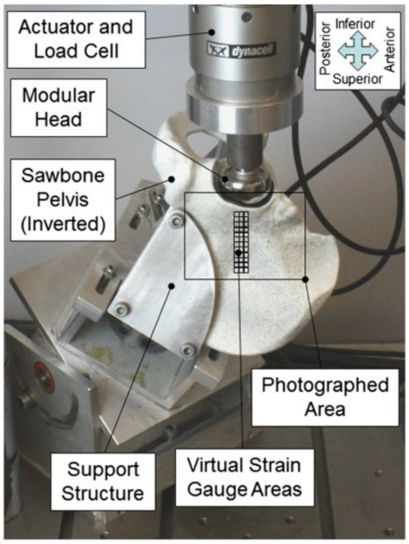


Figure 5: Recently developed setup for testing a hemi-pelvis where the pubic joint is restrained[2]

[1] Reprinted from [35] with permission from Elsevier

When evaluating the inter-fragmentary movement several methods have been used. For measurement, a displacement transducer (linear variable differential transformer, LVDT) [37] may be used. Optical means are also implemented where two pictures are taken from the front view and side view, and then the 3D displacement of the parts is calculated. The other technique is using non-colinear infrared light-emitting diodes (LED). They should be attached to each side of the crack, and by an optoelectronic camera system the 3D coordinates of the markers may be measured to find the gap opening [23]. In addition to these methods, digital image correlation (DIC) is another method that could be used to evaluate inter-fragmentary movement. Looking at the inter-fragmentary movement with DIC is a novel aspect of this thesis work.

[2] Reprinted from [38] with permission from Elsevier

III. The principles of Digital Image Correlation (DIC)

When a camera takes an image of a surface covered with speckles, it saves the data as discrete values for each pixel. Depending on the light intensity of each point, a number is saved on the memory to recreate the image on the monitor using these numbers. A sample intensity pattern of a 1D speckled surface is shown in Figure 6.

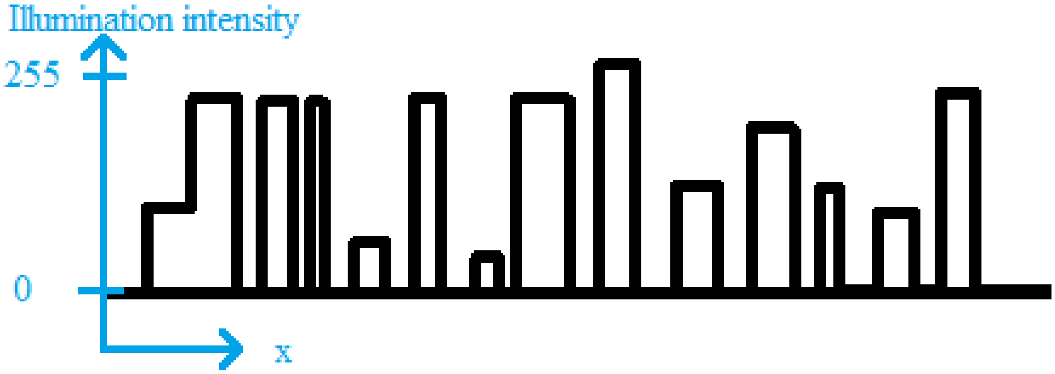


Figure 6: A sample intensity pattern of a 2D speckled surface[3]

The illumination intensity of a point can be from zero to the maximum capacity of the camera (analog to digital (A/D) converter's capacity). For example a camera with an 8-bit A/D converter can record the illumination up to 255 counts. With the random speckled surface, a random intensity pattering can be created as can be seen in Figure 6. The ideal pattern should be isotropic meaning that it should not have a preferred orientation. It also should be non-periodic [43]. Therefore, it is best to have a random pattern of small speckles on the object.

Having two images of the same object before and after movement and/or deformation, finding a single pixel again in the second image only by its gray value (illumination intensity) is

[3] The figure is taken from [44] with permission from Springer Science and Business Media

impossible as there could be many points found with the same value. However, if a neighborhood can be found with the same pattern just like or very similar to the neighborhood in the first image, it would be possible to detect the displaced pixel in the second image. Using two images of the object an algorithm is needed to track and find a certain point of interest from image 1 in image 2. So a subset should be defined around the point with an arbitrary size as shown in Figure 7.

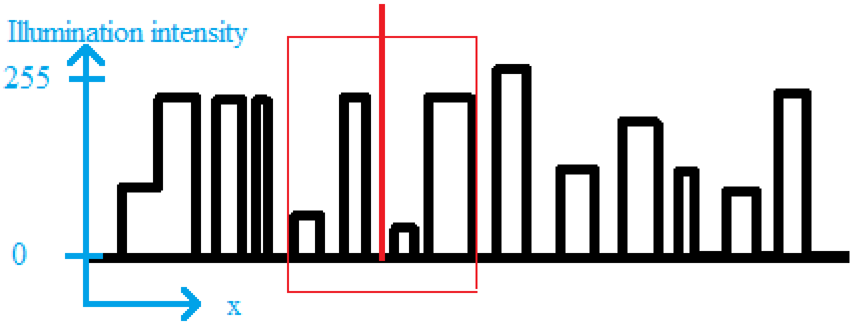


Figure 7: A selected subset with a unique illumination pattern

Now the algorithm must be looking for a subset with the same pattern in the second image. Although it is very easy for a human to track the subset displacement during the process and find it again in the second image, it is difficult to run a mathematical algorithm to do so. There are different methods for image matching such as those based on differential methods [43]. It is notable that due to noise and the difference in the lighting of the object, or because of deformation, the patterns in the images are not identical. In order to measure the similarity of the patterns, the subsets should be approximated by a substitution pattern (shape function) which is less complicated enabling the program to run a comparison algorithm. This simpler model can be a zero order, bilinear, or a higher order model.

To find the correlated subsets, one must find the most similar subset in the second image. For example, by sum of square residual and an iterative approach, it is possible to find the most similar subset in the second image. One optimized technique is named the squared sum of differences (SSD). The other method, from which the name of "digital image correlation" is taken, is normalized cross-correlation criterion. This method compares the initial subset to all the subsets in the second image and allocates a number to each (the green line in Figure 8). Finally, the subset with the highest cross-correlation value is considered as the moved subset and the displacement ΔX is recorded as the displacement of the middle point P.

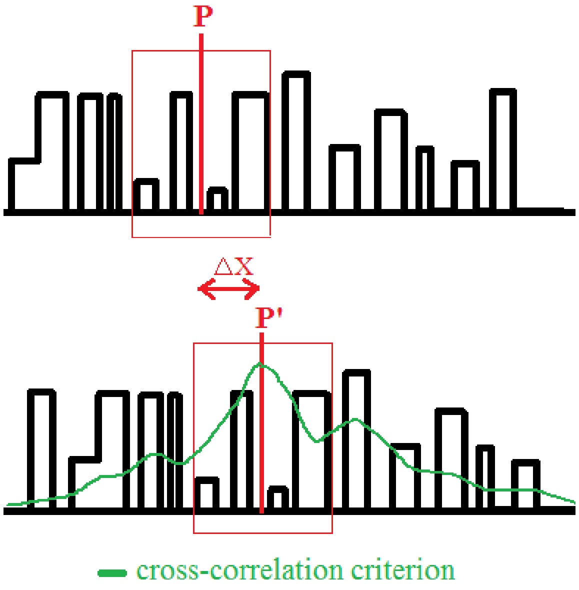


Figure 8: Using the cross-correlation criterion to track the subset

The number of pixels in the subset must be odd to enable the program to choose the center point. Assuming that the subset size is small enough, it can be assumed that the sides remain as lines after deformation.

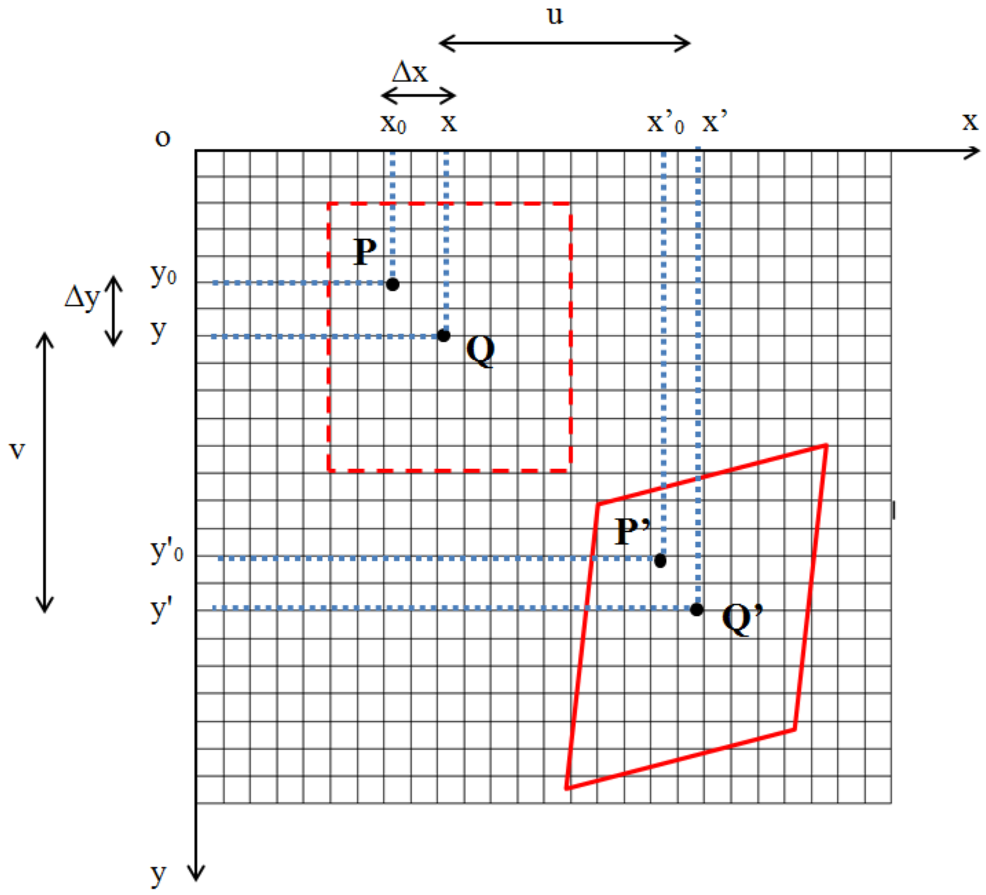


Figure 9: The initial and following status of the subset around the point Q

Having found the new coordinates (*x', y'*) of a chosen point *Q* (*x, y*) within the subset after the displacement and /or deformation, it can be shown that the governing equations are as follows:

$$x' = x \quad + \Delta x + u + \frac{\partial u}{\partial x}\Delta x + \frac{\partial u}{\partial y}\Delta y$$

$$y' = y \quad + \Delta y + v + \frac{\partial v}{\partial x}\Delta x + \frac{\partial v}{\partial y}\Delta y$$

where u and v are the displacements in the x and y directions for the subset center point P, and Δx and Δy are the distances from point $P(x_0, y_0)$ to point $Q(x, y)$. Reversely, knowing the coordinates of a few points such as *Q'*, it is possible to find the terms u, v, u_x, u_y, v_x, v_y. With this technique, it is possible to derive the displacement fields of *you* (*x, y*) and *v* (*x, y*) throughout the region. Also the strain field on the surface of the material in the point $P(x_0, y_0)$ can be found by:

Normal strain in the x direction: $\varepsilon_x = \frac{\partial u}{\partial x}$

Normal strain in the x direction: $\varepsilon_y = \frac{\partial v}{\partial y}$

Shear strain: $\varepsilon_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$

Further details about tracking algorithms have been discussed in [43]–[47].

IV. Employing DIC for Strain Measurements

There are a number of methods to measure the strain and mechanical properties of materials; however, they have some limitations. For instance, strain gauges can only measure the average strain over a finite number of areas. They could possibly miss areas of interest where strain concentration could be found because the usual amorphous geometry and material inhomogeneity in biomaterials very often result in sharp strain variation across a structure. This inhomogeneity is because of the fact the bone is comprised of cortical and trabecular tissues; moreover, it has a graded structure which results in varying properties throughout the length of the bone [48]. The other issue is that strain gauges need to be attached directly to the specimen. This is usually a wet contact with fresh or hydrated materials which could cause potential problems since the strain gauge should be attached directly to the surface of the material. Regardless, quite a number of papers using this technique [49]–[51]. Some other alternative techniques for measuring mechanical properties such as tensile tests or three-point bending test are destructive and at the same time have the same restriction of measuring finite points and wet conditions. Also, some of them might increase the local stiffness of the material.

Photoelasticity is capable of measuring the strain field over a large area of interest, but the limitation of this technique is that a photoelastic polymer material is required for which production of a replica structure is needed, and it is mainly used for 2D in plane testings. In case of a 3D geometry photoelastic coating may be implemented in the surface of metal specimens. Plus, excessive specimen curvature which can be seen commonly in bio materials cause problems in this technique [52].

Optical techniques, which have been developed recently, are gaining attention. This is because of the fact that they are very accurate as well as can be implemented at different scales. An interesting optical technique is digital image correlation (DIC) which will be discussed in details in the following section.

One of the major strengths of DIC is that it can be applied to complex and irregular geometries that are often found in bio materials and tissues. This flexibility of measuring biomaterials has made it quite popular in the recent studies [53], [54]. For example, it is used to investigate potential fracture criteria of rat femora. With a compression testing of rat femora, DIC has been implemented for a full-field strain measurement in the linear elastic range and close to the fracture event [55]. Similar work can be found on bovine femur focusing on cortical tissue [56]. In dentistry studies, this technique was chosen to measure the strain distribution and displacement on lower jaw when having occlusal loadings [57], [58].

DIC has been vastly used for different application in different scales of measurement such as micro or nano scale characterization of materials and bio tissues. DIC has been combined with low-power light microscopy strains at the post yield regions at the scale of 10 μm to 1 mm to study the inhomogeneity nature of the strain in elastic and inelastic deformation of bovine fibro lamellar bone [59]. Other studies have also used it to be coupled with optical microscopy and measured full-field surface displacement with micro scale accuracy [60]. Also, with a cellular scale on the cortical bone, DIC is used to investigate the strain filed around micro cracks [53]. Various and suitable speckle patterns corresponding to each scale have been studied for this purpose [61].

With nanoscale accuracy DIC is also combined with imaging techniques such as scanning tunneling microscope and atomic force microscopy. Using the scanning tunneling microscope imaging technique, it is applied to study the fiber/matrix interphase response in different composites [62]. Coupling DIC and atomic force microscopy led to measuring with an accuracy of 5 nm, over a 10 μm field of view to determine the elastic properties of poly-silicon [63]. Nonetheless, the pattern that they used for illumination intensity was based on the surface roughness profile of the sample while, as it will be explained, in DIC a speckle pattern using paint is commonly produced. However, other works can be found employing fluorescent nanoparticle tracers to produce the required contrast in images [61].

Because of the good precision and variety of usage condition of DIC, it is widely used for human bone, implants and prosthesis studies, for instance, to investigate changes in cortical strains before and after total elbow prosthesis [64]. The other application can be found in validating FE models to evaluate femoral neck fracture and periprosthetic stress shielding in the proximal femur after hip resurfacing arthroplasty [52], [65].

Finally, DIC has also been used for pelvis bone. DIC has been used to investigate the effects of hip implantation with cups made from different materials on changes in peri-acetabular strains [38]. They tried to investigate if using acetabular cups made from materials with closer elasticity modulus to cortical bone will lead to less adverse bone adaptation than cups made of stronger materials. Nonetheless, other applications of DIC in biomedical studies there can be found [54], [66], [67].

With the same fundamentals, modified techniques have been proposed to address the adverse demand for measuring mechanical properties in other shapes of tissues such as blood vessels having complex geometries. Although DIC itself was used before to investigate strain changes during stent inflation *ex-vivo* of arteries[68], 360˚DIC method or panoramic-digital image correlation (p-DIC) is proposed and used to characterize arteries obtained from mice less than 6mm in length and less than 1mm in diameter [69], [70].

V. Objectives

As discussed in the previous sections, no study can be found evaluating pelvic fixation and the fracture of a broken pelvis using DIC. Nonetheless, the fundamental tools are there to create a testing apparatus for such tests. As previously mentioned, the motivation of the current work is to investigate the strain field on a region of interest of a broken hemi-pelvis as well as measure the inter-fragmentary movements. This will be accomplished with the following objectives:

1) Determine the worst case scenario of the femur force orientation based on the everyday activities.
2) Modify the existing testing apparatus (MABTA) and create jigs to test a hemi-pelvis.
3) Calibrate and validate the load cell and motor of the testing apparatus in order to measure force displacement.
4) Develop a program to measure gap opening in 3D using DIC.
5) Evaluate the repeatability and accuracy of measuring the gap opening and strain field using DIC.
6) Measure the strain field on an intact hemi-pelvis using DIC.
7) Measure the strain field on a fractured hemi-pelvis using DIC.
8) Measure the gap opening (inter-fragmentary movement) of a broken hemi-pelvis in 3D using DIC.

Chapter 2: Experimental Setup

This chapter describes the experimental setup. Section I covers an introduction to MABTA as the main frame of the setup. It will discuss the controlling program as well as the specification of the load cell as an important device for measurements. As the setup needed to be adapted for the purpose of testing of pelvises, a new part was designed to make sure that the pelvis is placed at a defined angle. Section II presents the procedure of finding this angle and implementing it. The equipment used for implementing the DIC technique and the description of its main software can be found in Section III. A custom-made program written in Matlab will also be discussed at the end of this section.

I. Setup (MABTA)

The multi-axis biomechanical testing apparatus (MABTA) (Figure 10) is a testing apparatus developed in 2006 in the Mechanical Engineering Department at the University of Alberta [71]. MABTA was designed as a multi-use apparatus. Originally this particular design was to enable the user to examine either complete or partial cadaveric human, animal or composite knee specimens [71], but generally, its quasi-static loading frame may be used to conduct *in vitro* experiments on different organic or synthetic bones.

Furthermore, as this biomechanical testing system is designed to allow working on cadaveric human and animal specimens, sealed bearings, stainless steel and aluminum have been used wherever possible to allow chemical sterilization of the parts without damaging the integrity of the design.

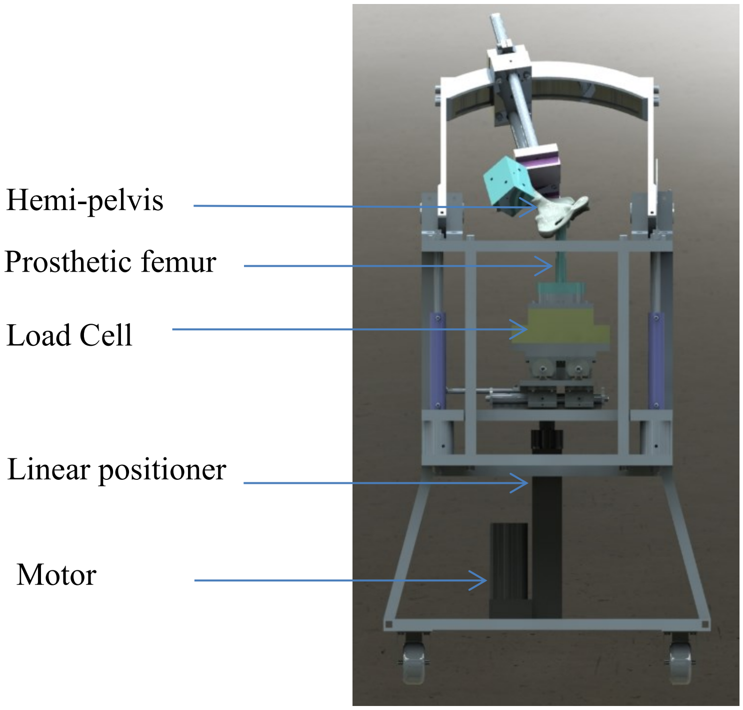


Figure 10: Annotated figure of MABTA

MABTA is assembled on a painted stainless steel frame with casters to allow mobility. The largest aspects are the width of 0.762 m (30”) and the height of 1.828 m (72”). These dimensions allow MABTA to fit through a typical door and be easily moved. When designed, by utilizing finite element analysis and hand calculations, the stiffness of its frame was estimated at 7321 N/mm (41806 lbs. /in) [71].

MABTA features a 6-degree of freedom support for testing samples using a vertically moving plate. Between the moving plate and the other stationary support, there is a six-degree of freedom load cell (its specification can be found in section 2.I.B) to measure forces and moments in three directions.

Loading and moving in the vertical direction is achieved by using a 150 mm stroke IDC Motion EC3-B electromechanical actuator and Kollmorgen servomotor (Model AKM42G-BSCN2-02). The actuator has a backlash of ± 0.25mm and a repeatability of ± 0.013mm. A linear positioner converts the rotational movement to linear movement as can be seen in Figure 10.

To use a 3D DIC system on the MABTA, two cameras were needed. The specification of the cameras will be explained in Section III. Assembly of the MABTA with the cameras can be seen in Figure 11. This assembly was used for a test on the hemi-pelvis which will be explained in Chapter 4.

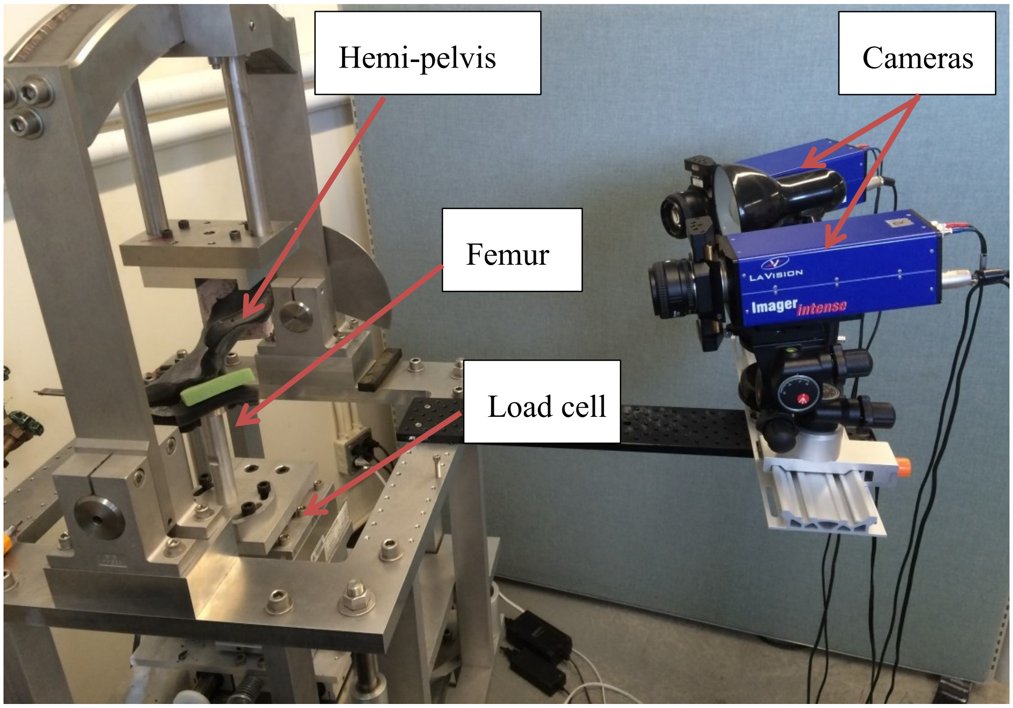


Figure 11: The MABTA assembly with the cameras to run a DIC test

i. MABTA Control

Both the servomotor and the load cell are connected to a computer via serial (RS-232) port. Therefore, a graphical user interface (GUI) program was needed to communicate to the ports. Using commercial software (LabWindows/CVI) an initial CVI code was developed in Dr. Nobes' group at the University of Alberta to control the machine for displacement up to a critical load coupled with a Matlab program. The Matlab program is also written in Dr. Nobes' group at the University of Alberta to control the cameras (the section 2. III will discuss the cameras) connected to the computer. The set up was used to do some preliminary tests. After initializing the main program, it sends a command to the Matlab program to start taking pictures with the cameras. The program is capable of applying simple compression or tension loading until the load reaches a certain defined value. Having reached the limit, the motor stops, and the program sends a message to the Matlab program to stop taking pictures. The main panel of the program can be seen in the Figure 12. The main panel of the Matlab program can be found in Figure 13.

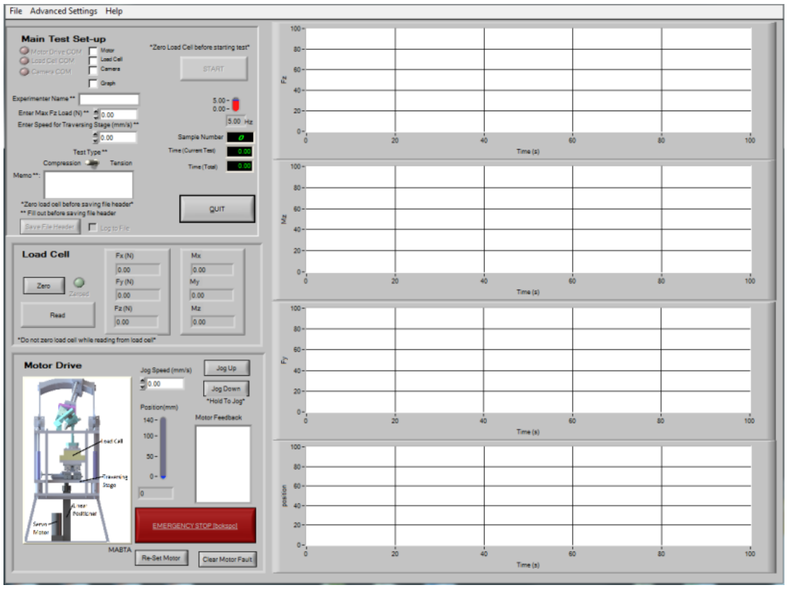
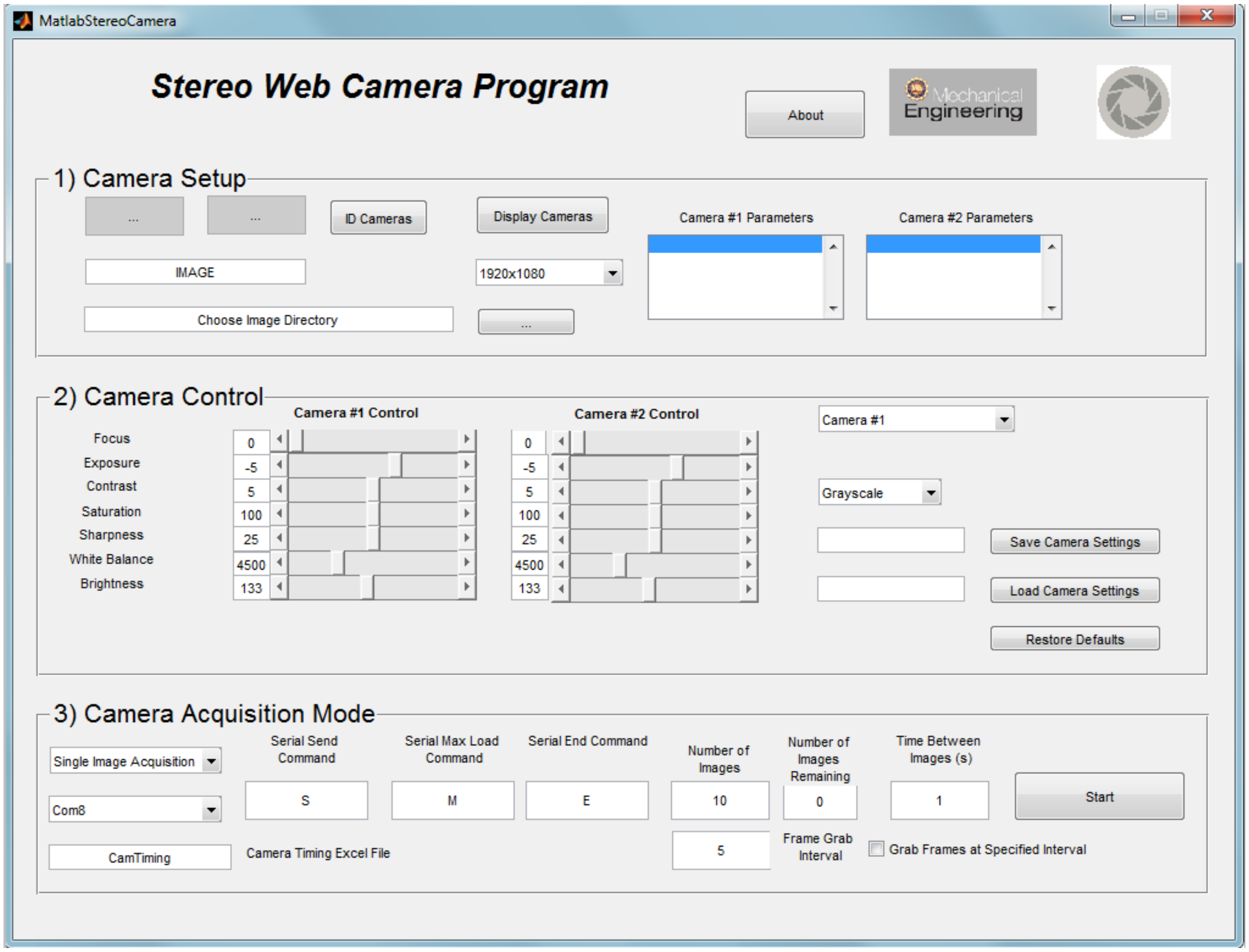


Figure 12: The main panel of the old version of MABTA program (MABTA V.2)



IMAGE

1920x1080

	Camera #1 Control	Camera #2 Control
Focus	0	0
Exposure	-5	-5
Contrast	5	5
Saturation	100	100
Sharpness	25	25
White Balance	4500	4500
Brightness	133	133

Camera #1

Grayscale

Single Image Acquisition

Com8

CamTiming

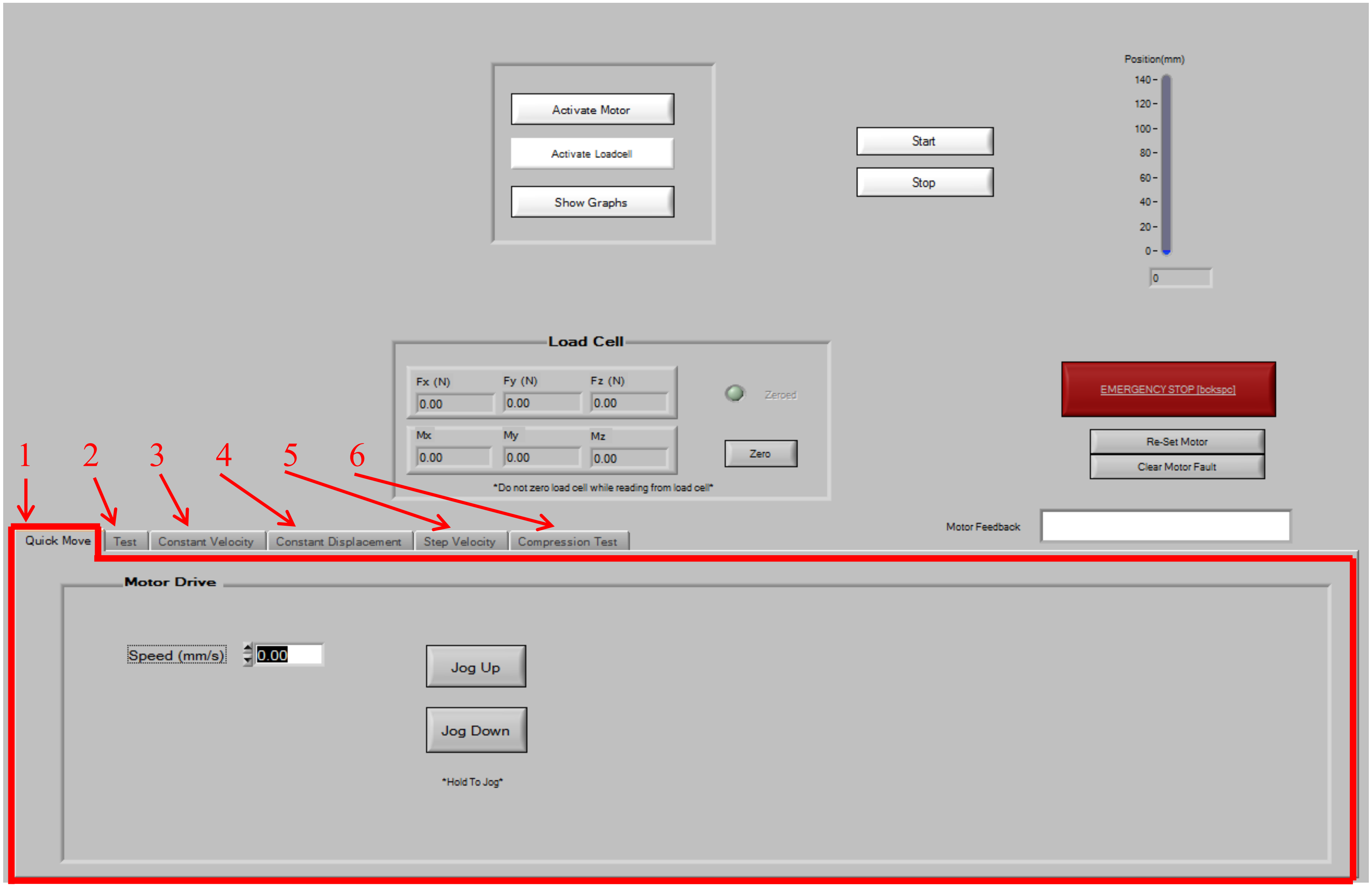
Serial Send Command	Serial Max Load Command	Serial End Command	Number of Images	Number of Images Remaining	Time Between Images (s)
S	M	E	10	0	1

Frame Grab Interval: 5

□ Grab Frames at Specified Interval

Figure 13: The main panel of the GUI program for taking pictures with webcams

The program was modified to include more features, and to be more user-friendly. The most important modification of the program was changing the settings to get a higher resolution of its position. The position of the moving plate was reported in terms of millimeters earlier, and it needed to be of higher accuracy. So by changing the setting of the motor, the resolution in micrometers was achieved. Moreover, the old features of the older program were split into two tabs. As can be seen in the Figure 14, a "Quick Move" tab was created to move the motor up and down. The CVI code of this program can be found in the appendix III.



Position(mm): 0

Fx (N): 0.00 Fy (N): 0.00 Fz (N): 0.00

Mx: 0.00 My: 0.00 Mz: 0.00

Speed (mm/s): 0.00

Figure 14: The main panel showing the "Quick Move" tab (MABTA V.3)

The version one features were moved to the second tab: “Test”. This tab can be used to get coupled with the Matlab program, and it does a simple compression test. This tab is the tab 2 shown in Figure 14. Another tab was added to move the motor at a constant speed for a certain period of time. This tab with the name of “Constant Velocity” is the 3rd tab shown in Figure 14.

Another tab was created to have a constant displacement. There is a button to specify the motor speed as well. This tab is shown by number 4 in Figure 14. Also, a step movement tab was added to the program as the 5th tab in Figure 14. The 6th tab can be used for a compression or tensile test without coupling to the Matlab program.

ii. Load Cell

The main controlling program connects to an AMTI signal conditioner (MiniAmp MSA-6) (Figure 15) used to measure the loads from the AMTI 6 DOF load cell (MC6-6-2000). The model is suitable for simultaneous measurement of several forces and moments. The applied loads are resolved into orthogonal force and moment components over time. The capacities of the load cell can be seen in the Table 1 and Table 2.

This load cell is constructed from high strength T7075-T6 aluminum alloy. The sensors, which are installed inside the load cell, incorporate special seals to prevent water and oil contamination. These features make it suitable for working on cadaveric tissues and organic materials.

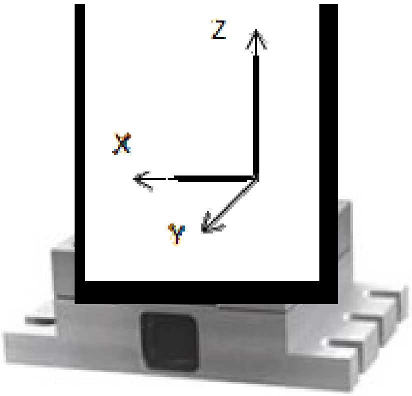


Figure 15: Load cell and the defined coordinate system for Table 1 and Table 2

Table 1: Load capacity of the load cell

Force	Capacity (lb.)	Capacity (N)
F_X	1000	4450
F_Y	2000	8900
F_Z	1000	4450

Table 2: Moment capacity of the load cell

Moment	Capacity (in*lb.)	Capacity (Nm)
M_X	6000	680
M_Y	6000	680
M_Z	3000	340

II. Pelvic Loading Orientation

i. The Background

If a pelvis is to be tested on an apparatus, boundary conditions such as the loadings (both magnitude and direction) and supports need to be defined. In case of biological parts and tissues, the natural conditions are often sought. With these conditions it can be possible to have an *in-vitro* testing of which results are quite similar to *in-vivo* testing, and consequently, similar to an actual situation. Therefore, the type of loading, its direction, and its magnitude have to be determined. The same argument can be made for the supports.

This section discusses what potential candidates may be introduced as loading conditions. A criterion for choosing between those candidates is proposed. Based on the proposed method, the final boundary condition is chosen, and is implemented for developing the apparatus.

To better understanding the loadings on the pelvis, one can look at Figure 16. It shows the forces exerted to the pelvis during walking. There is a vertical force on the sacrum because of the weight of the upper body. The left femur needs overcome this force in a way that the resultant force is toward interior to make the body move forward. Moreover, a dynamic movement requires a dynamic loading. If the sacrum is assumed as the support of the pelvis, the loadings on the pelvis can be studied by the femur loading.

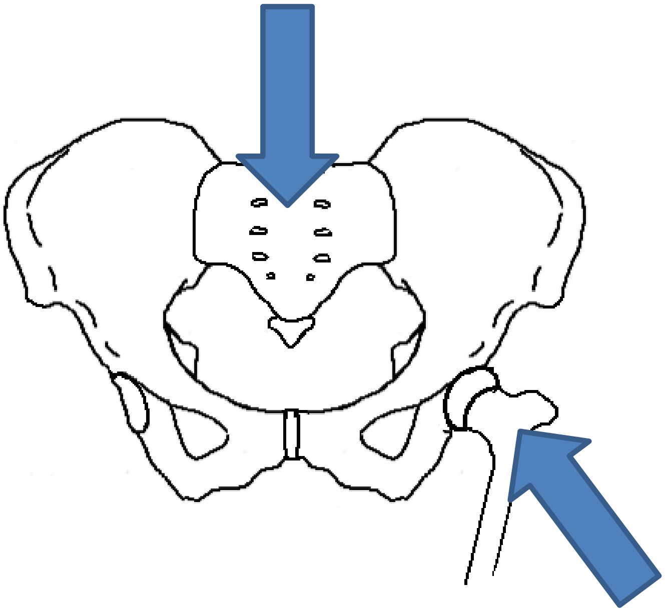


Figure 16: The loadings exerted to the pelvis during walking.

In other studies on the loading on the femur and pelvis [1] it is indicated that the highest values of loading can be seen at dynamic movement. Figure 17 shows the force directions exerted to the left hemi pelvis by the femur during some routine activities. As the result, the highest magnitude of femoral force was found to be 260% of body weight (BW) which frequently occurs in walking down stairs. In other activities less amount of force is exerted to the pelvis; however, these forces have different directions. Therefor the effect of these forces could be comparable to a force with higher magnitude, but acting in a direction on which pelvis might resist stronger. Thus, as developing a testing set-up, the most efficient loading condition is yet to be defined. No study was found investigating “the optimum” or “the worst case scenario” loading condition for this matter.

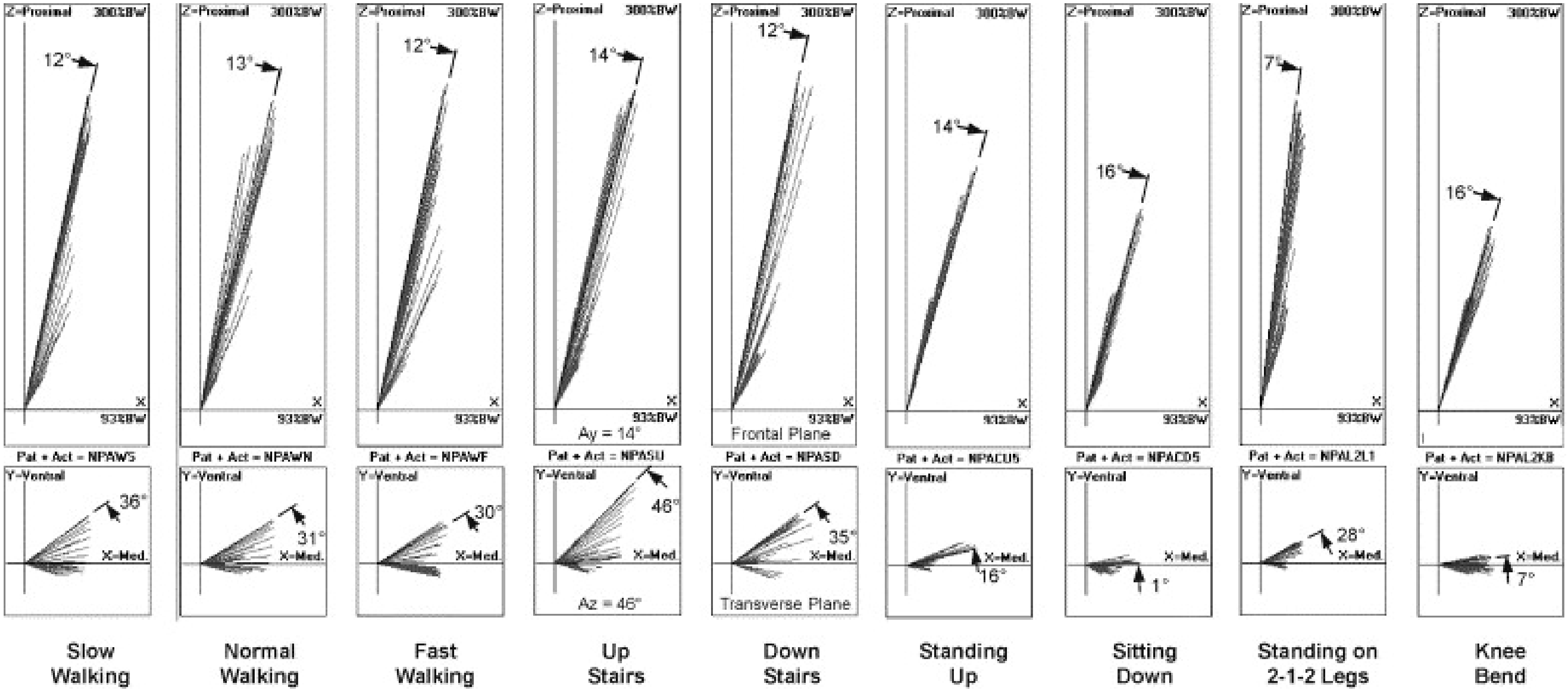


Figure 17: Loadings exerted to the pelvis in routine activities[4]

Note that the axes were chosen according to the standardized method described in section 1.

[4] Adapted from [1] with permission from Elsevier

ii. Simulation of Pelvis Loading

To investigate the worst case scenario for testing the pelvis, a finite element analysis (FEA) is done. The objective of developing the FE model is to simulate different loading scenarios and determine which one has the greatest stress. A male hemi pelvis was scanned, and a Computer-aided drafting (CAD) model was obtained. This model was imported to commercial software (SolidWorks, Dassault Systemes). The material properties were chosen as 17 GP for Young's modulus and 0.29 for Poisson Ratio. These values were found in previous studies [40] for cortical shell.

When specifying the boundary conditions, the inner surface of the acetabulum cup was loaded. The magnitude and the direction of the force were changed for different runs based on the aforementioned angles. As the fixed contact area, the sacrum surface was selected as the support for the simulation. However, these assumptions are not what would be precisely seen in an *in-vivo* testing. The reason is that the cartilage tissue in the femoral head does not create a uniform distributed pressure on the acetabulum face. Indeed, this ideal boundary condition, for exerting the force on the acetabulum, has been used in some studies before [35], [72], [73]. It can be achieved by using specific resins very similar to human cartilage and a realistic femoral head (either synthetic or cadaveric). Also, due to low mechanical properties of the pubic symphysis and considering the fact that the pelvis has very small movement toward this tissue, the force exerted by this tissue is often neglected. Moreover, the weight of a typical human was chosen as 80 kg. Therefore the magnitude of 2000 N as 250 %BW was used in the simulation.

The most critical candidates for testing a pelvis can be seen the Table **3**. The parameters: α and β are angles of force components along the transverse and coronal planes. These components are illustrated in Figure 18.

Table 3: Four sets of angles chosen to simulate the experiment [1]

	Magnitude (%BW)	α°	β°	Description
1	250	12	30	Fast walking on level ground
2	251	14	46	Walking upstairs
3	260	12	35	Walking downstairs
4	231	7	28	2-1-2 legged stance

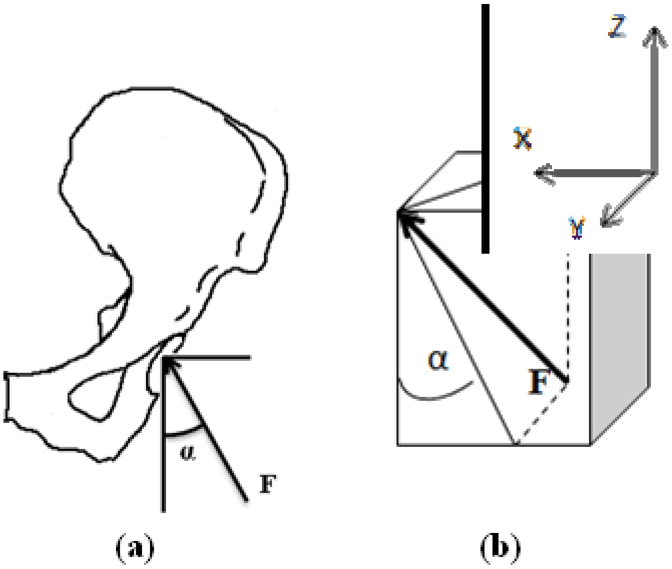


Figure 18: Force components. Figure 11.a shows the angle on the coronal plane and the force on the left hemi-pelvis, 11.b shows the angles on the coronal and transverse plane, and the 11.c shows the coordinate system.

As the mesh study, the maximum stress found in the model was investigated. Although the final result of the simulation was the displacement, but since the stress is the more sensitive parameter in simulations, the maximum stress in the model was investigated. the model was meshed by a coarse size of 14 mm and then the mesh size was reduced in seven trials to 8 mm. Figure 19 is obtained by the Trend Tracker tool in the SolidWorks, and shows that mesh size smaller than 13 mm leads to a stable result. All the loadings were simulated with the mesh size of 7.6 mm as can be seen in Figure 20.

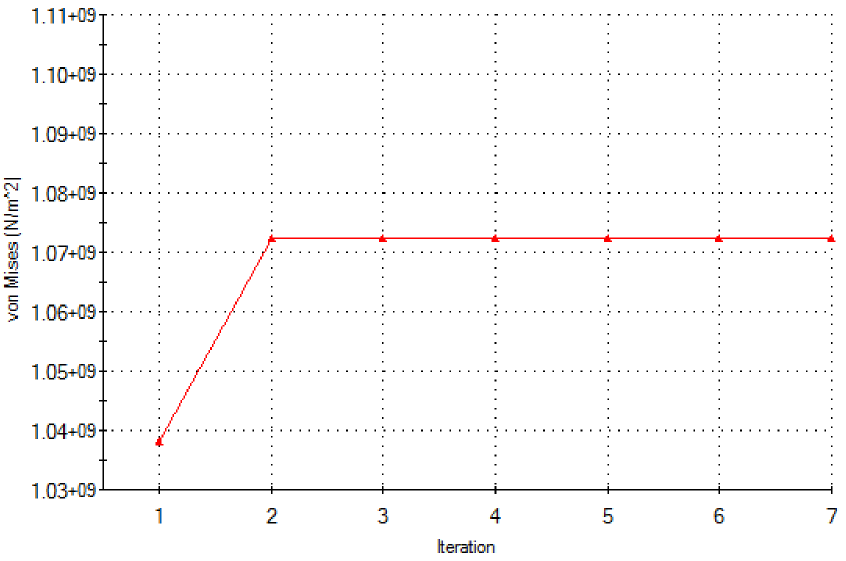


Figure 19: The mesh study showing the convergence of the von Mises stress with varying size from 14 to 8

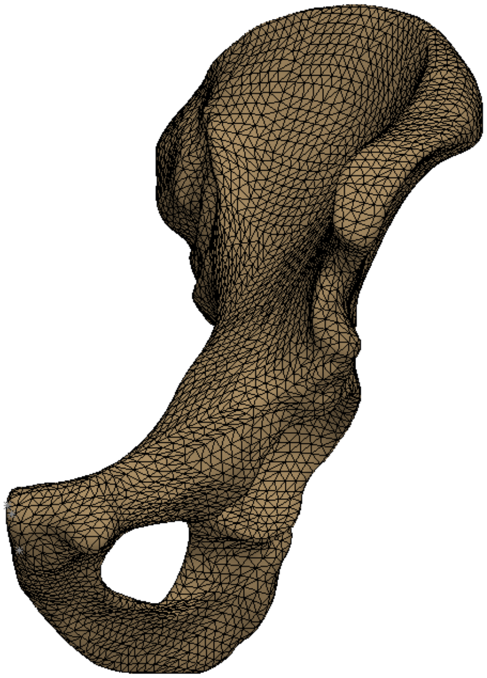


Figure 20: Meshed hemi-pelvis used in the simulations

The maximum displacements corresponding to different loadings can be seen in Table 4. It can be concluded that the maximum displacement in pelvic bone is found when walking downstairs. The color map of displacement field and von-Mises stress can be seen in Figure 21 and Figure 22 respectively.

Table 4: The results of the simulations

	Description	Maximum Displacement (mm)
1	Fast walking on level ground	1.47
2	Walking upstairs	1.58
3	Walking downstairs	1.63
4	2-1-2 legged stance	1.36

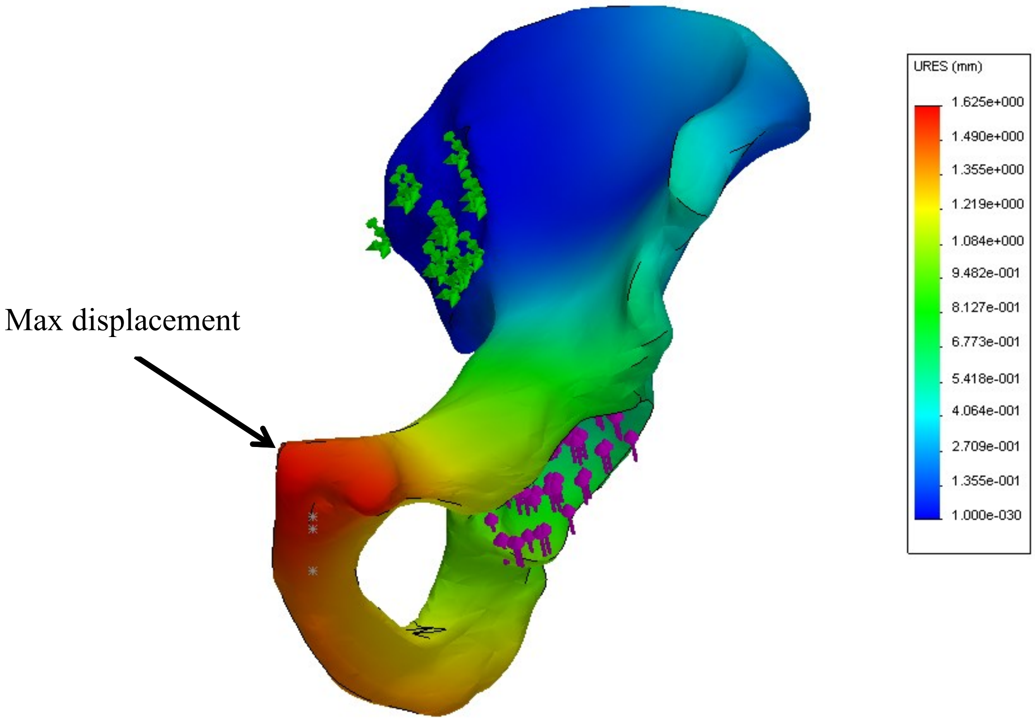


Figure 21: Displacement distribution. The loading is of type 3 (Walking downstairs)

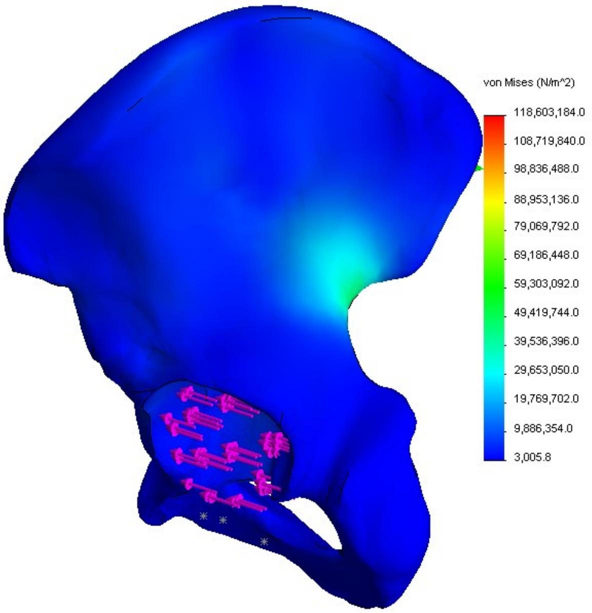


Figure 22: Von Mises stress distribution (Force type 3)

iii. Desired Angle

As can be seen in Table 4, walking downstairs withe pelvis3 mm deformation of pelvis is the most rigorous routine activity. This conclusion could have been somehow expected as it involves the maximum magnitude of femoral force. This finding should not overshadow the effect of loading direction. This effect can be seen when comparing the walking upstairs result to the value obtained for fast walking on level ground. Walking upstairs leads to higher deformation and strain, even though its force magnitude is as same as walking on level ground. As an explanation, the loading with $\beta = 46^\circ$ causes pelvis to undergo greater amount of moment the other does; whereas the loading with $\beta = 30^\circ$ exert a more axial load. As anticipated, the maximum displacement can be seen in the vicinity of pubic symphysis since it is the most distal point of the bone from the sacroiliac joint.

Finally, for developing the testing setup, the angles of $\alpha = 12^\circ$ and $\beta = 35^\circ$ should be used. If having a step loading to simulate routine activities, the force magnitude of 260 BW, which corresponds to walking downstairs, is suggested.

iv. Design of Potting Jig for Optimum Angles

Aforementioned discussion about the optimum angles for testing requires finding the anatomical planes on the hemi-pelvis. Using the angles, the hemi-pelvis has to be potted. The objective of this section is to design a jig that will allow for the pelvis to be potted at the optimum angles. So with the vertical force of prosthetic femur, the loading would be applied at the optimum direction. Once the potting is done with the satisfying orientation of the hemi-pelvis, it is possible to proceed. The procedure of determining the anatomical plane on the hemi-pelvis needs to be followed every time that a new hemi-pelvis is going to be tested. Therefore, having a jig in which the hemi-pelvis can sit is required. The other benefit of designing this mold is that the potting material needs time to dry and harden. As can be seen in Figure 23, it is very difficult to hold the sample still until the potting material dries.

Having found the anatomical planes on the pelvis, they were added to the CAD model of the scanned hemi-pelvis (Figure 24). Then, the hemi-pelvis and the mold were assembled using the angles which had been obtained before. This assembly can be seen in Figure 23.

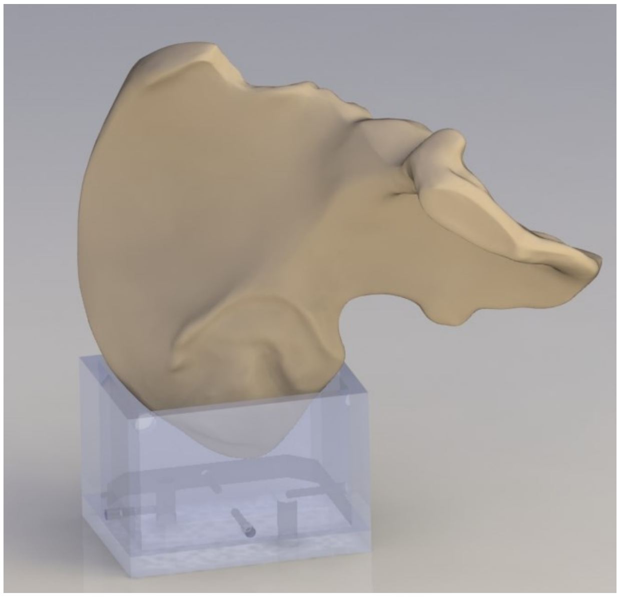


Figure 23: Hemi-pelvis and the pot assembly

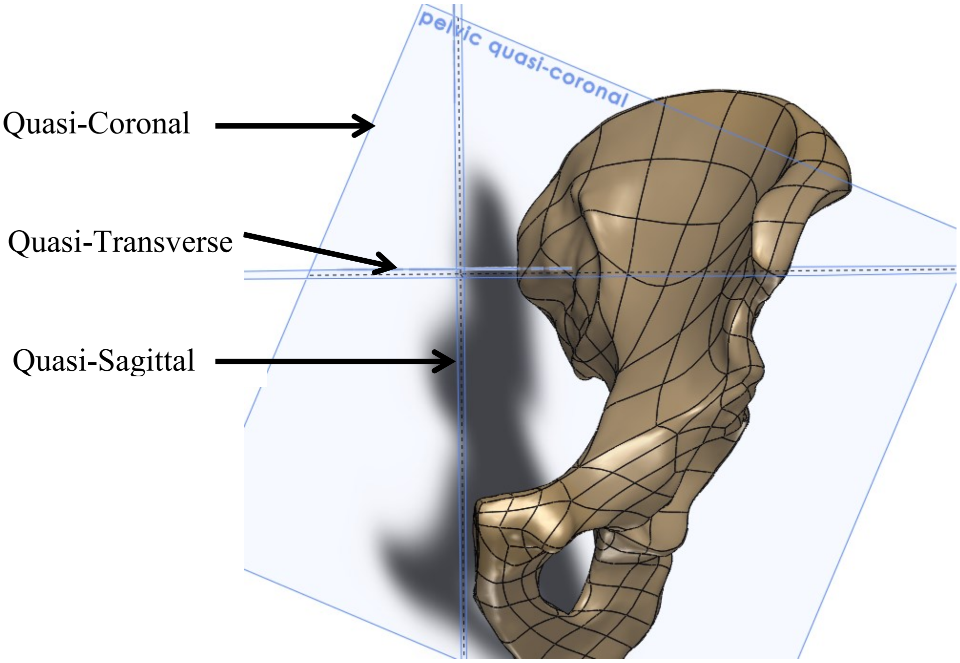


Figure 24: Anatomical planes defined on the pelvis in SolidWorks

At the next step, a mold was designed to hold the hemi-pelvis at the shown position. The mold consists of two pieces, allowing it to be removed after potting the part. Due to the complex shape of the hemi-pelvis and consequently the mold, it was needed to have the mold manufactured by 3D rapid prototyping. The mold can be seen in Figure 25.

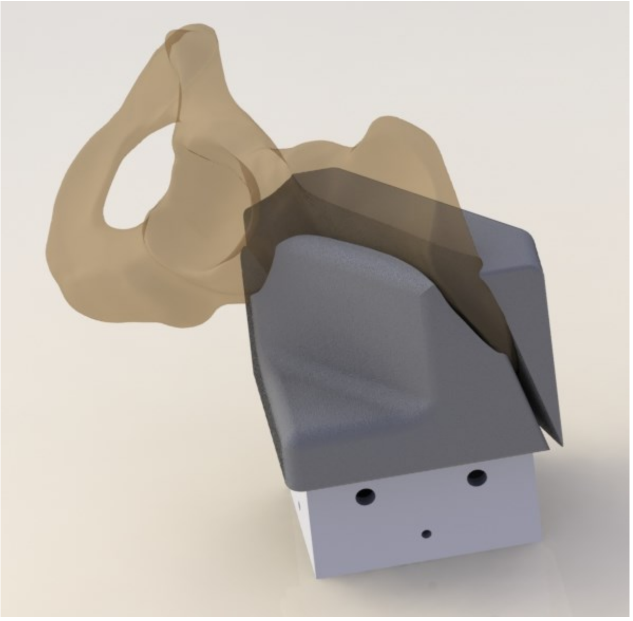


Figure 25: CAD model of the potting assembly

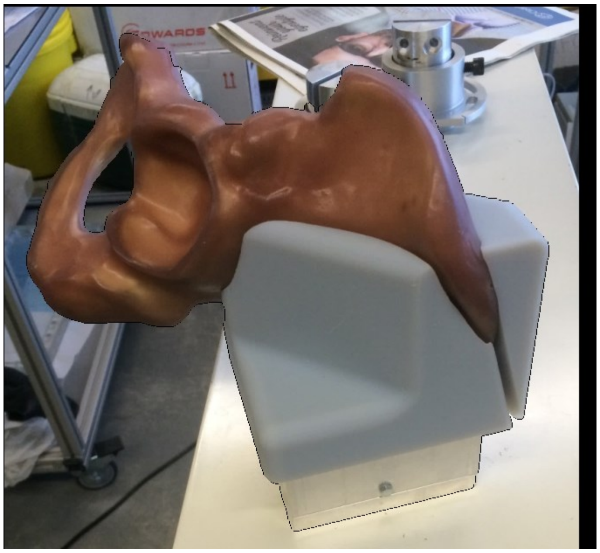


Figure 26: The real assembly after manufacturing the mold

III. Digital Image Correlation (DIC)

i. Data Acquisition

As the main data acquisition device, two sets of cameras were used for this project. In the early stages of the project, Microsoft webcams (2 megapixels) were used. Figure 27 shows the webcam. The resolution of the pictures was 1920 x 1080, and the analog to digital converter's capacity was 8 bits. The cameras were controlled by the Matlab program that was explained in Chapter 2. In order to use the entire possible range of illumination (up to 255 count), and get a good contrast in the images, a feature was added to the Matlab program. This feature enables the user to see the illumination intensity of any points when previewing images. The other important issue is the exposure, which could have been controlled by the Matlab software. The main limitation of the program is the time lag between capturing pictures with the cameras. Usually there is a 0.2 sec lag for capturing the second image with the other camera. Due to this problem, it is not suggested to have a dynamic test using this program.

Figure 27: Microsoft webcam[5]

[5] picture is pulled out from www.microsoft.com

In later stages of the project, LaVision cameras were used. One of them was "Imager Intense" and the other was "Flow Master 3S (FM3S)". The cameras were fairly similar with the resolution of 1280 x 1024 pixels for FM3S and 1376x1040 for Imager Intense. Figure 28 shows the "Imager Intense". The analog to digital converters' capacities of both of them were 12 bits. The cameras could be controlled by the same commercial software that the company has developed for image processing (DaVis). The other necessary controlling features have been included in the software such as exposure time, finding the illumination distribution of images, and controlling the contrast of the pictures. This software is also capable of capturing pictures with both cameras exactly at the same time, which makes it possible to have a dynamic test with the setup. Since the resolutions of the cameras are more than necessary to perform the tests, the small difference between the cameras would not be a problem.

Figure 28: LaVision camera (imager intense)[6]

[6] picture is pulled out from www.lavision.de

ii. Image Processing and Obtaining the Strain Field

Having some images from the speckled part undergoing displacements or deformation, the pictures should be processed. After processing it would be possible to interpret the movement of the particles on the surface. In many cases, this image processing is performed using commercial packages. More details about the procedure will now be explained.

During the work, when multiple pictures were taken from the test piece (for example, Figure 29), they needed to undergo an image processing algorithm to calculate the displacement field. The commercial software that was used for image processing was LaVision DaVis v.8. In the early stages of the project when calibrating the camera system was more difficult (because of the lower precision of the webcams), some pre-processing functions were used. However, most of the images were processed without any pre-processing function. As explained before in the principles of DIC section (Chapter 1), for finding the displacement and strain of a center point, a subset must be selected. The software allows the user to determine the size of this subset. The subset size of 65 pixels was reached after trying different sizes. Smaller subsets result in indistinguishable and inaccurate strain field. On the other hand, although the bigger subsets would result in a smoother strain field, it is not so different from the suggested 65x65 pixels results; moreover, much more computation resources are demanded to have all the processes done in this way. As an example, to perform the processing with 85x85 pixels subsets, the 8 GB RAM of the computer was not sufficient. Also the results of using 71x71 pixels subsets were similar to 65x65 pixels subsets. Of course, the suggested subset size is found for particle size of ~20µm and the pixel size of ~10µm. Other subsets may be required for a different speckle pattern and pixel size. A sample displacement field can be seen in Figure 30.

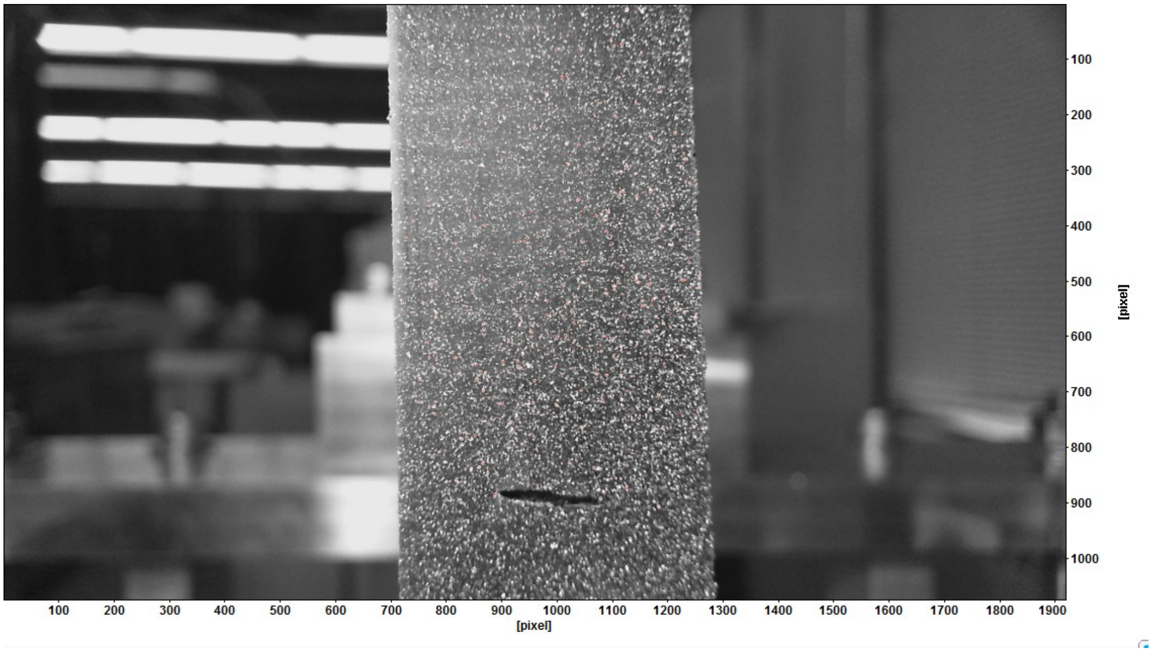


Figure 29: The raw image of a test piece. This image was the 6th image after the tensile test of the rubber.

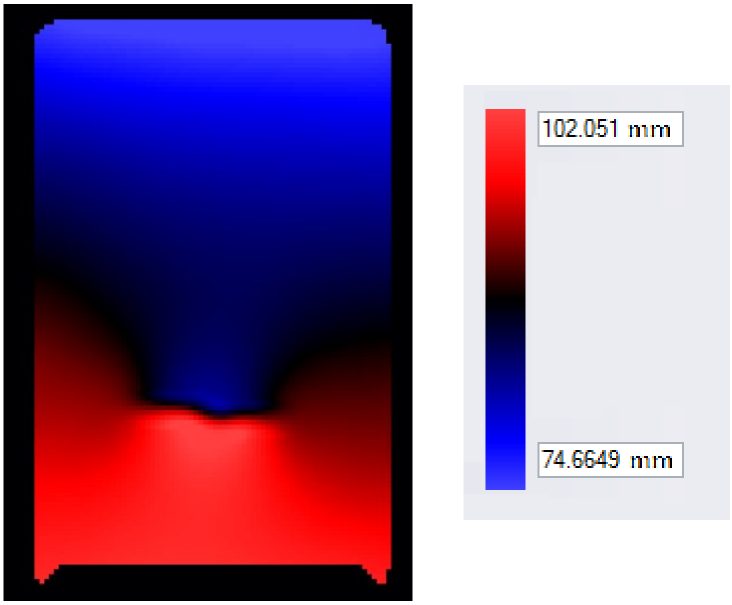


Figure 30: Vertical displacement field of the test piece

The other issue is that it is not necessary to calculate the strain for every point with the distance of 10μm because there is not a dramatic gradient of strain within the material. Therefore, a feature can be found in the software to determine the desired distance between the points for which a calculation has to be done. This value was usually determined from 2 pixels to 8 pixels. In most cases 4 pixels as the distance was chosen.

DaVis is also able to find the strain field with a high order of precision and other internal advanced filters. A sample strain field can be found in Figure 31. It allows the user to identify the region with a higher strain than the other areas. This strain field can be plotted. For this matter, an area of interest must be selected in order to find the average strain over the area. The output of the software is the average strain on the area for every picture. It simply allows the user to follow the strain variation during the experiment. This area could be as small as just a few pixels though.

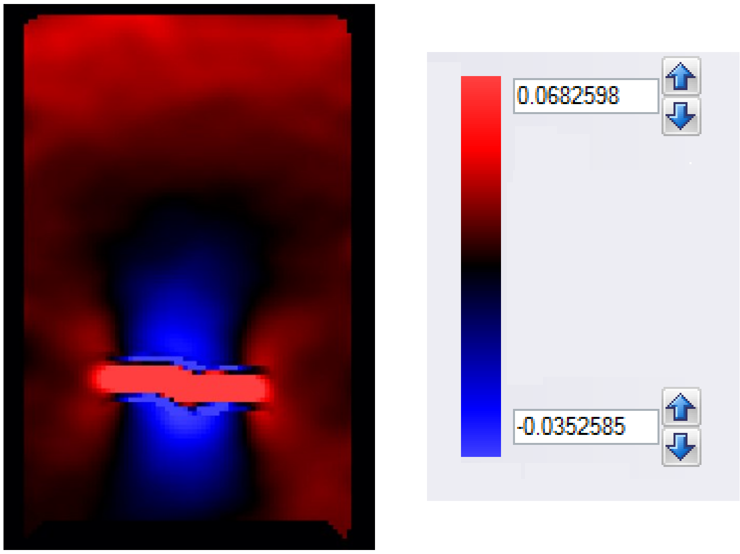


Figure 31: Strain field (Eyy) of the test piece

iii. Gap Measurement

When evaluating fracture fixation it is imperative to measure the gap opening. The cameras will allow the user to measure the gap in every step of the experiment without using a different device which needs to be very accurate to find the displacement of certain remarks on the both sides of the gap.

DaVis does not give the displacement of the arbitrary points during an experiment; however, it does calculate the displacement of every point in the process. After having the images processed to determine the strain and displacement, this information is saved in the format ".VC7" in the directory of the project.

In order to read this file and import it to Matlab, the LaVision GmbH company has released a Matlab function: "load_3DVC7()". After addressing the vc7 file, the function creates several variables, including the displacements of each point in three directions.

For naming the points, the first image is the reference, meaning that if a point with coordinates of [20 , 10] is the 18th point from the left and the 33rd point from the bottom, in all "vc7" files this very point (which is still the 18th from the left and the 33rd from the bottom in the matrix) is named [20 , 10] even though it has moved to a new coordination.

This issue makes some complications. For instance, assume one would like to see the information corresponding to a point which is at the coordinates [5 , 5] in image 10, but is not possible to just simply find the information of the point [5 ,5] since this is the information of the point which was on [5, 5] initially. So it is needed to open the "vc7" file of the first image and find the corresponding row and column of the point [5 , 5] and then find the cell with that very column and row in the displacement matrix of the 10th image again.

The other issue is that DaVis creates a grid on which it selects the points for calculating the movements and strain. However, once the displacement of a point is desired, it may not be found on the data simply because the point does not lie on the grid as can be seen in Figure 32. Therefore, a substitute on the grid should be selected to carry on. Since the grid size is a few pixels (the grid size can be specified in the software with the minimum of 1 pixel) and the pixel size is considerably small, the subset is very close to the desired coordinates.

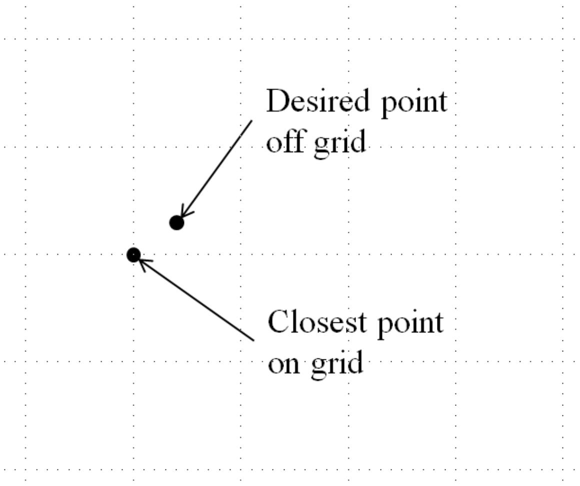


Figure 32: Having a point off the grid and the substitute on the grid

Considering the aforementioned issues, a Matlab program “gapmode.m” was developed. The program gets the address of the file created by DaVis as well as the locations of the two desired points and the step number at which the gap opening is sought. Then it finds the closest points on the grid and finds the displacements of the points form the initial position up to the desired step. Adding up the displacement and the initial position, the new position can be found. Finally, the Euclidean distance between two points can be derived. The program can be found in Appendix I.

iv. Opening Mode

One important feature of the gap opening is the mode of the opening. Since the bone toughness is different in the mode I and mode II (Figure 33) of crack opening [74], [75], it is important to determine on which mode the gap opens. This issue is important since it is preferable to have the gap opening on the first mode on which the bone shows its maximum resistance [74]. Thus, a better fixation should restrict the movement of the broken part on other modes as much as possible.

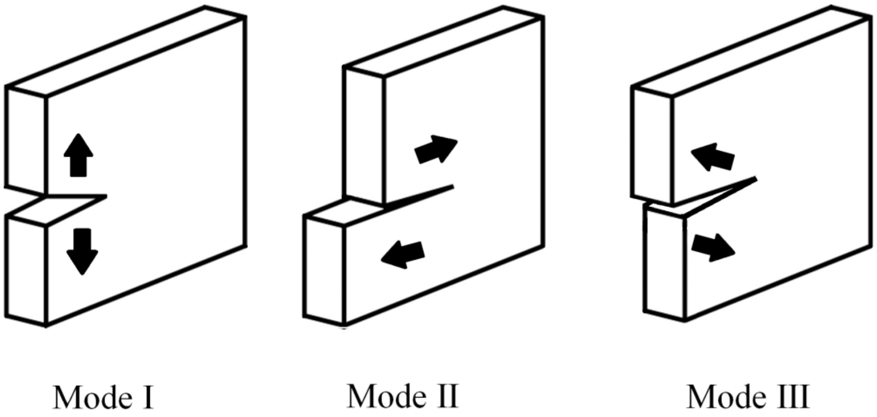


Figure 33: Crack modes

So a Matlab program "gapmode.m" was developed to determine the gap opening mode. The program defines a new Cartesian coordinate system by taking the coordinates of the new origin, a point on the x axis, and a point on the XY plane. Then it reports the 3D displacements of the two desired points based on the new coordinate system. The program can be found in Appendix II.

Chapter 3: Validation of the Setup

After combining the components of the apparatus, they needed to be validated. In the following chapter, the validation of the experimental setup is presented. In section I, the load cell is evaluated. After calibrating the load cell, the ability of the measurement system to report the load-displacement was evaluated by a compression test. In section II and III pure displacement tests can be found. The goals of these tests were to look at the ability of the motor to have a uniform and accurate movement (section II), as well as the ability of the visual setup to measure a uniform displacement field (section III). In section IV, the gap measurement program is tested to find its error, and section V evaluates the gap mode measuring program. In the next section VI, evaluating the strain field of a rubber piece was the goal. This experiment determined if the setup produces a repeatable result for the strain field. After getting a reliable strain field in this section, the strain field values obtained by the setup were validated in section VII. A tensile test on a standard dog bone test piece was performed to validate the strain field. After completing the previous steps, it can be said that the whole setup is validated. It is notable that tests on the rubber with large gradients of strain and displacement as well as tests on the dog-bone samples with small displacement validate the setup performance beyond the necessary accuracy required to test the synthetic hemi-pelvis.

I. Load Cell Evaluation with Compression Tests

To check the load cell accuracy, the load cell was calibrated using a Proving Ring, accurate to 0.025% or better of full scale (Morehouse Instrument Company). During the process, at different displacements, the values that the load cell showed was recorded and compared with the values that the proving ring showed. The test was done up to the limit of 2000 N which was the limitation of the proving ring. Table 5 shows that the maximum error was 1.24%, which occurred at 2014 N. The data are again represented in Figure 34. Figure 34 shows a linear behavior, and from the regression value of 0.99999 (R-square), it can be said that the error of the load cell is linear. Using the least square method the formula of the line was derived. The ideal formula would be $y=x$. The formula for the line was derived as $y = 0.99\ x + 3.41$. The load cell could have been calibrated again with this information, but due to the difficulty of changing its settings, all loads were reduced by 3.5 manually when recording. It should be noted that the accuracy of the load cell is ±2.5 N in the vertical direction.

Table 5: Load cell calibrating data

Load cell (N)	Spring (N)	% error
300	297.26	0.92
689	684.64	0.64
1023	1014.32	0.86
1316	1303.31	0.97
1607	1589.89	1.08
2014	1989.28	1.24

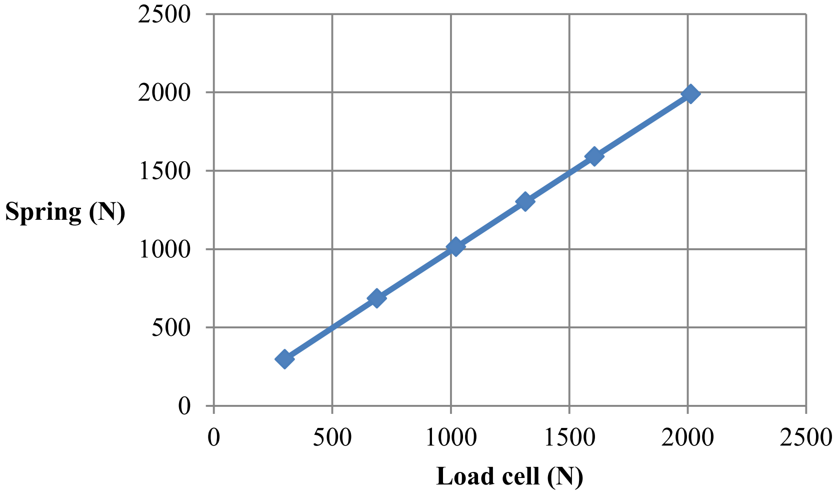


Figure 34: Spring force vs load cell force for calibrating

In order to ensure that the software is able to report the force-displacement data with a good timing (without any lag) and accuracy, a second compression test was designed. The material was a PMMA cylinder with the dimensions of 19 cm length and 3 cm diameter. The test piece was tested once using an MTS machine and again with the MABTA. The deformation rate was 0.5 mm/s for both machines. The tests were done up to 2500 N compressive force. Figure 35 shows the results. Although PMMA shows a slightly different behavior due to viscoelasticity, the curves have similar slopes. The curves are not totally overlapping because of the viscoelasticity effects. Nevertheless, the elastic moduli of the material, which can be interpreted as the slope of the lines, were very similar and within the range of PMMA which is 1.8-3.1 GPA [76]. Table 6 shows the comparison between the slopes of the curves in the interval of 500N – 2500N.

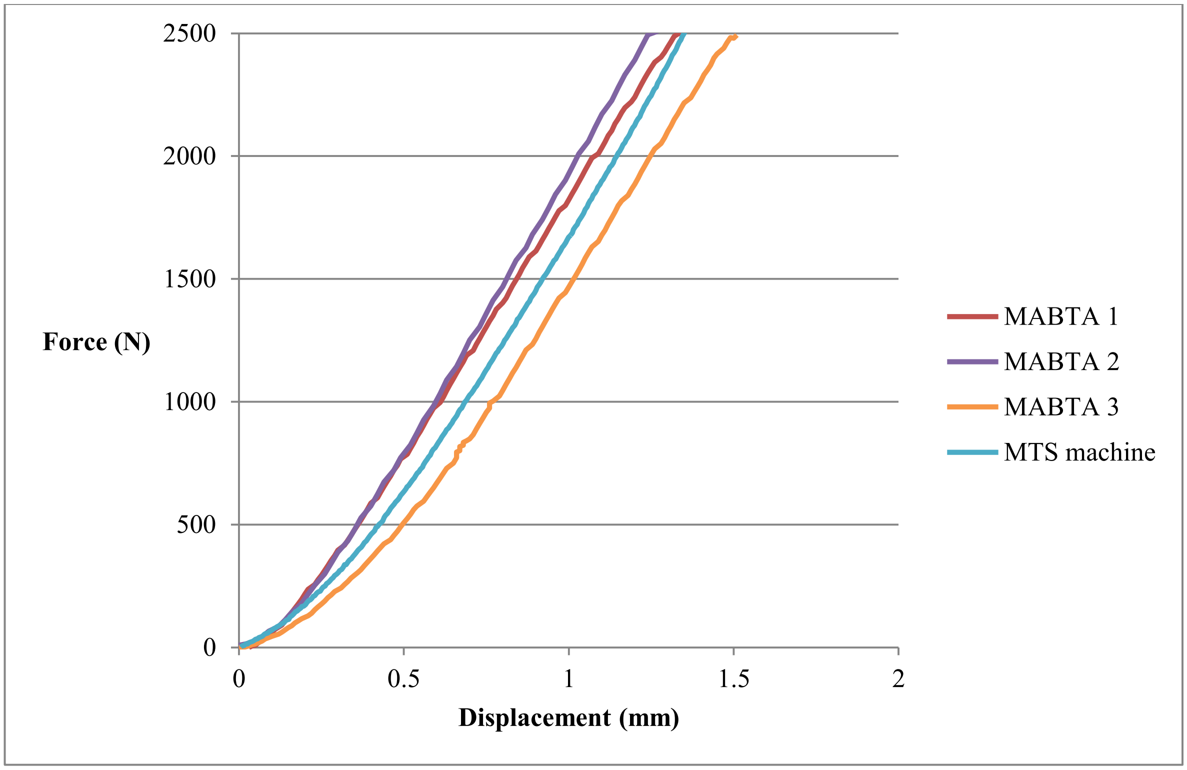


Figure 35: Load-displacement curve of the testing sample

Table 6: Calculated curvature slope for the compression tests

Test #	**Slope (N/m)**	**Average (N/m)**	**Deviation (N/m)**
MTS machine	2.1	-	-
MABTA 1	2	2.1	0.005
MABTA 2	2.1		
MABTA 3	2.2		

As shown, the load cell works with good accuracy and precision. It is also able to measure the force displacement data with a good precision and small deviation of 0.005. At the end it can be said that the load cell of MABTA maintains its accuracy during a varying load as well.

II. Pure Displacement Using a Dial Indicator

To make sure that the motor can move the setup at a constant speed over a defined time, it was calibrated. The objective was to find the curve showing the relation between the nominal and actual displacements for different velocities.

Calibration was done using a dial indicator with the accuracy of 0.001inchesh as can be seen in Figure 36. The displacement of the traversing plane was measured at known speeds and time. It was expected to have a small deviation. The measurements were done in 3 different levels to cover the whole moving height. To ensure that the dial indicator is perpendicular to the plane, two images were taken to make it parallel to the bars of the MABTA. Since the bars used in MABTA are parallel to each other, this idea can be used that two unparalleled planes corresponding to two parallel lines make a third parallel line. The images used for this purpose can be seen in Figure 37. Using the images, it was confirmed whether the dial indicator is parallel to the bars or not since it was difficult with the naked eye.

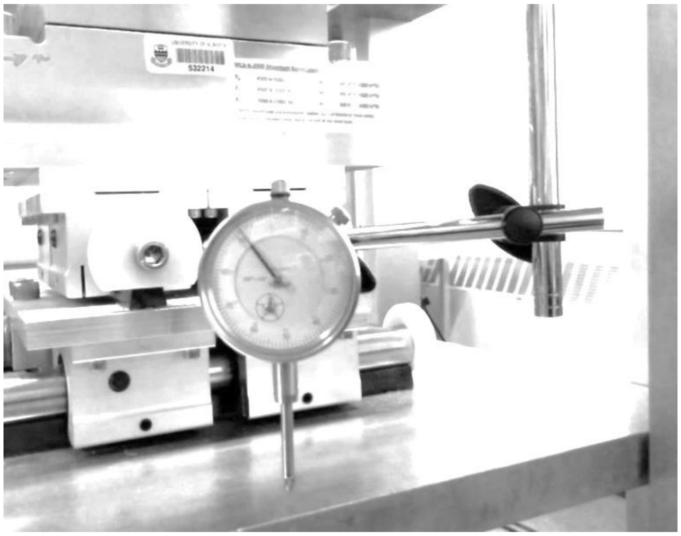


Figure 36: The set-up used for calibrating the movement

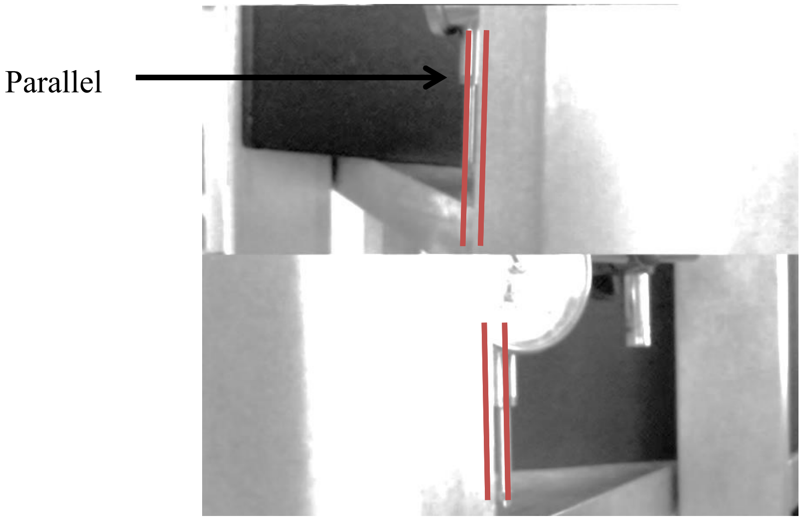


Figure 37: Images used to have a right angle

The tests were carried out for both moving up and down, which turned out to have similar results. The velocities at which the measurements were done were: 0.5, 0.7, 1, 1.5, 2, 3 mm/s, and the maximum displacement was 1 inch. For each velocity the increments were different as for smaller speeds, wider ranges of time were available. The numbers of increments are shown in Table 7.

Table 7: Number of increments for testing in each velocity

V (mm/s)	Number of increments
0.5	11
0.7	10
1	9
1.5	9
2	8
3	6

The motor also was tested under the load of 1000~1500 N to see if it keeps working with the same precision when loaded versus unloaded. As can be seen in Figure 38, a spring with a stiffness of 16000 N/m was used for this purpose. After testing 8 times at different velocities, the same results were obtained.

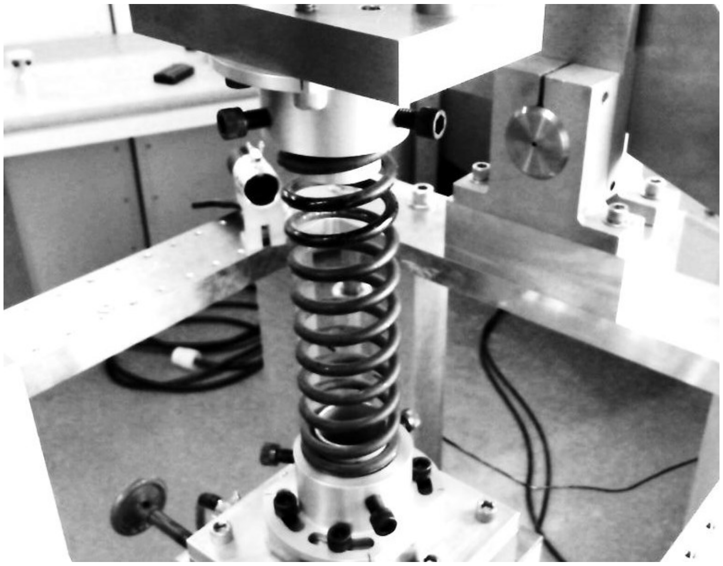


Figure 38: The set up with a spring

The results are illustrated in Figure 39 where the actual displacement was measured from the dial gauge while the measured displacement was recorded from the MABTA. It can be seen that the data are lying on the y=x line. The average error in displacement was 0.9% and the maximum was 2%.

Using the least sum of squares method, the following equation was obtained for the motor performance:

$$Y = 1.008\,X + 0.011$$

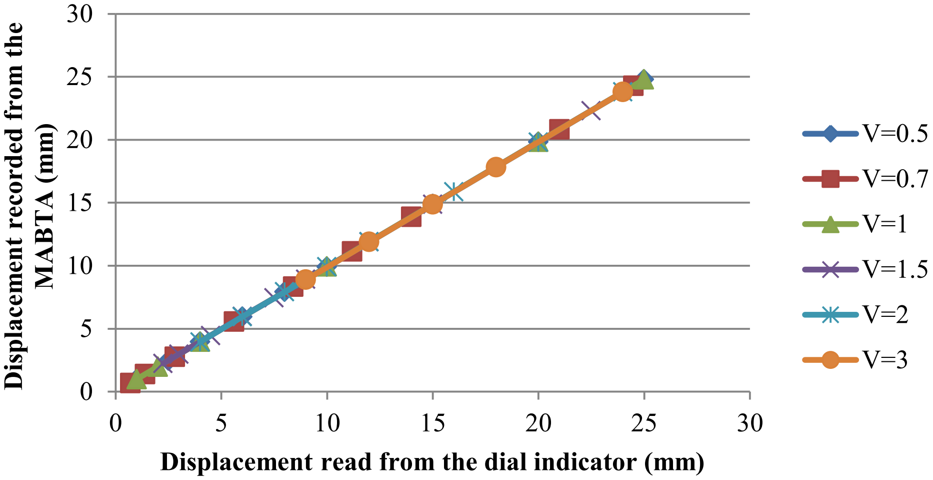


Figure 39: Results of pure displacement in loaded condition

It is noted that the motor was not able to work properly if running for less than a minimum time, so those results were removed from Figure 39 and are shown separately in Figure 40. For example, at the speed of 1.5 mm/s the motor should have moved 1.5 mm per second while in the very first second this value was less than 1 mm. This minimum of working time increased with increasing the velocity at which the motor was working. The minimum time for each velocity can be seen in Table 8.

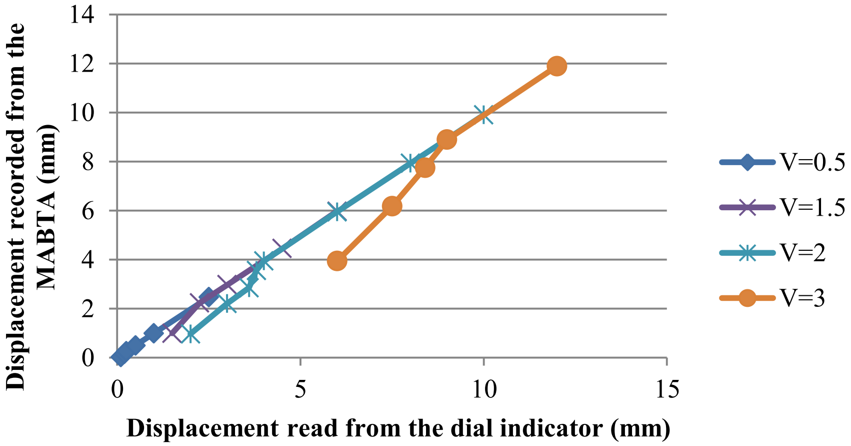


Figure 40: Deviation at initial time

Table 8: The feasible minimum time for each velocity

Velocity (mm/s)	**T_{min} (s)**
0.5	0.5
0.7	0.8
1	1
1.5	1.5
2	2
3	3

During the procedure and moving the plane up, when stopping the motor, the plane returned around 1mm. Also the motor could not hold the plane in that position when a large amount of force was being exerted to it. This was due to the program commanded the motor to be disabled after stopping, so the motor could not bear any loads when turned off. Therefore, by changing the code to keep the motor on all the time, this problem was solved.

III. Pure Displacement Using the Camera Setup

In the next step, to start testing the camera setup, another experiment was done. The purpose was to have pure movements in 3 directions. In the test, the cameras were set up. To use a commercial image processor software (LaVision DaVis 8), following the recommended procedure of the company, a pair of pictures were taken of a calibration target to find the position of the cameras related to the test piece. Then a plate which had a speckled pattern was installed on the MABTA. After having an initial picture using the Microsoft Webcams, the plate was moved. The movements were measured by the dial indicator with the accuracy of 0.001". After taking a photo from each state, the photos were processed using a commercial image processor software (DaVis 8, LaVision GmbH), to find the displacement field. The displacement which was found in each state can be seen in Table 9, Table 10, and Table 11. The coordinate system was assumed a right-handed Cartesian as can be seen in Figure 41

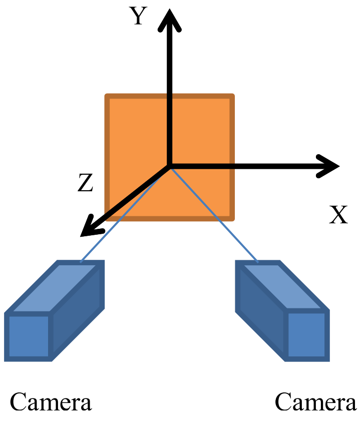


Figure 41: The global coordinate system

Table 9: Displacement error of a simple plate on the x direction

Displacement (1000* Inch)	Displacement (mm)	Measured displacement X (mm)	% Error
50	1.27	1.25	1.54
100	2.54	2.52	0.95
150	3.81	3.76	1.31
200	5.08	5.02	1.23
250	6.35	6.27	1.34
300	7.62	7.53	1.19
		Average:	1.26

Table 10: Displacement error of a simple plate on the y direction

Displacement (1000* Inch)	Displacement (mm)	Measured displacement Y (mm)	% Error
50	1.27	1.29	1.34
100	2.54	2.54	0.05
150	3.81	3.79	0.43
200	5.08	5.03	0.89
250	6.35	6.30	0.74
300	7.62	7.54	1.01
		Average:	0.74

Table 11: Displacement error of a simple plate on the z direction

Displacement (1000* Inch)	Displacement (mm)	Measured displacement Z (mm)	% Error
50	1.27	1.26	0.82
100	2.54	2.48	2.45
150	3.81	3.70	2.92
200	5.08	4.93	2.86
250	6.35	6.17	2.88
300	7.62	7.37	3.29
		Average:	2.54

The data have been demonstrated again in Figure 42. As can be seen, the error of moving perpendicularly to the cameras (z direction) is higher than the other two directions as expected. The error of moving is less than 2% for moving in *the XY plane* and less than 3.5% for moving in *Z* direction. The main source of error was the webcams which were not designed for this purpose and did not show the required precision. It was expected that when moving to the LaVision cameras, higher precision would be found as they are made for this purpose. The more accurate calibration values later confirmed the theory.

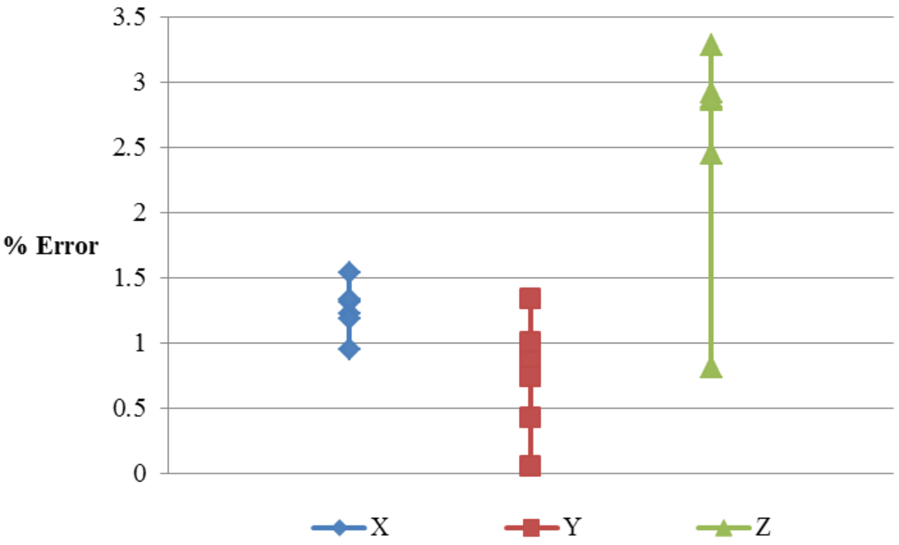


Figure 42: Displacement error of moving a plate in different directions

IV. Gap Opening

One of the goals of the work is to measure the gap opening between the broken parts. It is very important that we would be able to measure a gap in three dimensions. Using DIC and obtaining the displacement field of the sample, we need to measure the gap opening. As has been described in the chapter 2, a Matlab code “gap.m” was developed to calculate this parameter.

To validate the accuracy of the gap measurements, an experiment was designed. In the test, two plates were attached to a caliper. The caliper and the plates can be seen in**Error! Reference ource not found.**. After calibrating the camera system (the LaVision cameras seen in Figure 43 and Figure 44) initial pictures were taken.

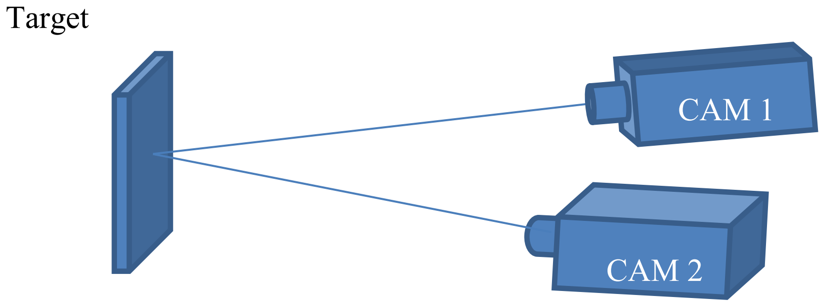


Figure 43: Schematic of the cameras positions

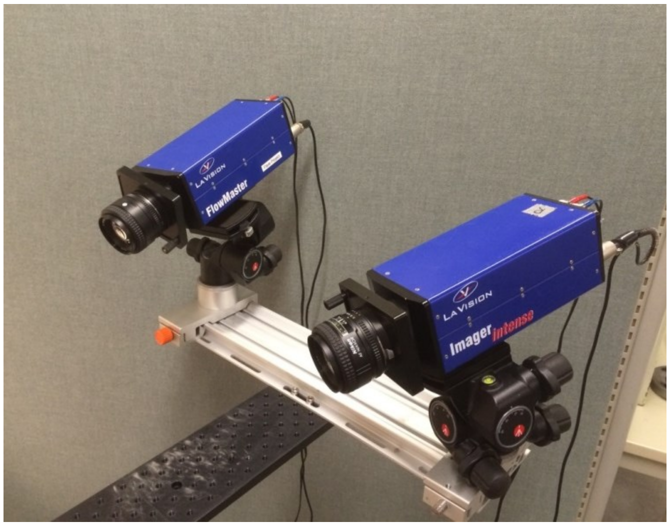


Figure 44: Cameras positions

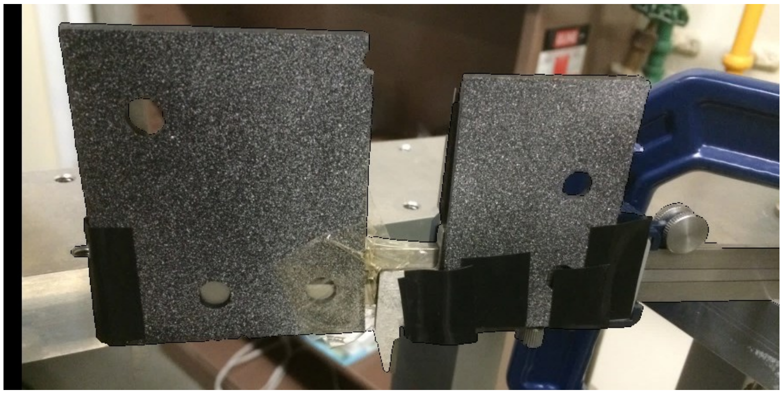


Figure 45: Caliper and plates assembly

Then the distance between the plates was changed as can be seen in Table 12, and then another picture was taken from the position. The positions were at 0, 0.2, 0.4, 0.6, 0.8, and 1 mm representing an in-plane opening. The accuracy of the caliper was ±0.005 mm. The caliper may not have been the best tool with 5 μm accuracy, but it was the only available tool for this matter. Four movements were detected between each state. The whole procedure was repeated 3 more times. Then after processing the picture in the commercial image processor software (DaVis 8, LaVision GmbH), the distance between the plates at every state was driven and compared with the values read from the caliper. The data can be seen in Table 12. The data are represented again in Figure 46.

Table 12: Testing data for gap opening

		Measured opening (mm)					Average (mm)	Deviation (mm)
Test 1	Measured	0.206	0.195	0.201	0.204	0.212	**0.204**	**0.000164**
	% Error	0.6	0.5	0.1	0.4	1.2	0.6	**0.00634**
Test 2	Measured	0.212	0.201	0.192	0.2	0.191	0.199	**0.000283**
	% Error	1.2	0.1	0.8	0	0.9	0.6	**0.0104**
Test 3	Measured	0.199	0.207	0.205	0.185	0.201	0.199	**0.000324**
	% Error	0.1	0.7	0.5	1.5	0.1	0.6	**0.0144**
Test 4	Measured	0.198	0.199	0.197	0.198	0.192	0.197	**0.000034**
	% Error	0.2	0.1	0.3	0.2	0.8	0.3	**0.0034**

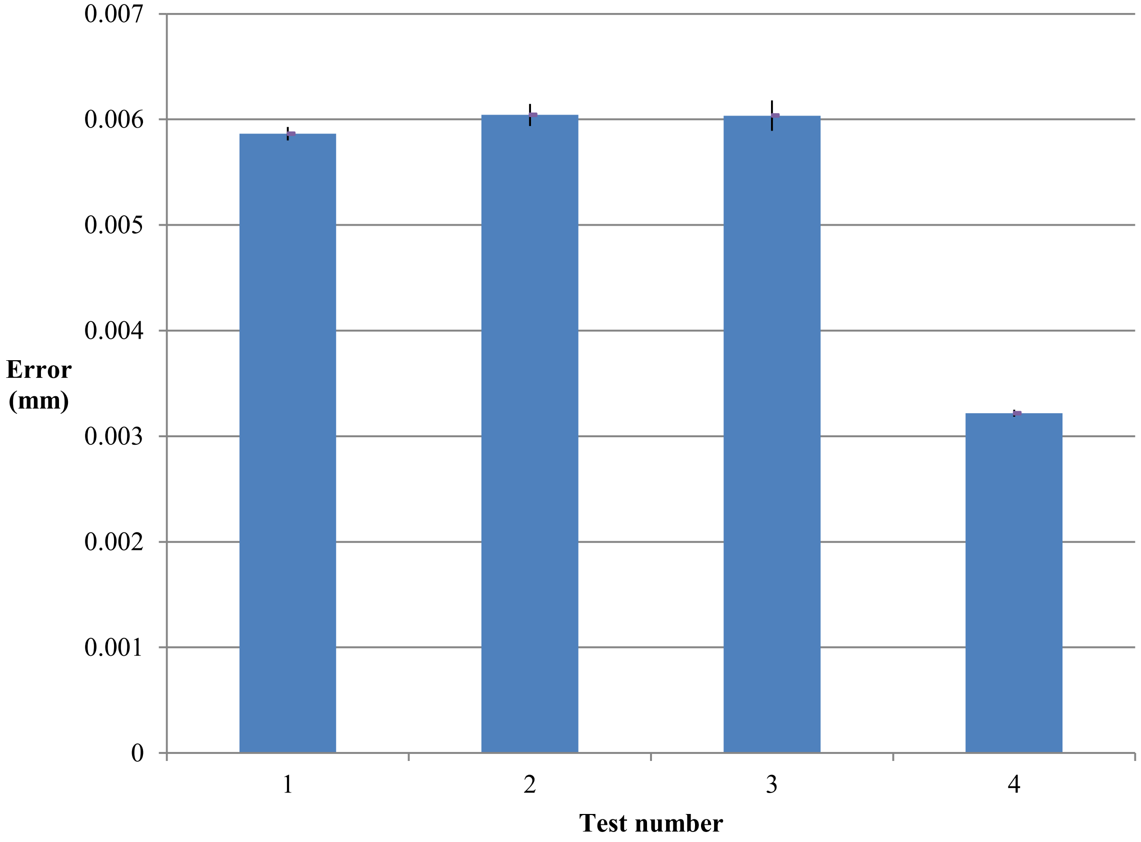


Figure 46: Average error and its deviation for gap opening

As can be seen in Table 12, assuming that the accuracy of the caliper was 0.01 mm, since the maximum error was 0.015 mm, it can be said that the accuracy of the visual system is 5 µm. Likewise, if assuming that the average error of caliper was 0.005 mm, since the maximum average error of the visual system was 0.006 mm, it can be said that average accuracy is 1 µm. Also, another test was done for moving toward the cameras which will be shown in the following Section. The Accuracy of moving toward the cameras was found as 30 µm.

It can be said that the custom-developed gap measuring program can calculate the gap accurately. When it is coupled with the visual processing system, a highly accurate measurement tool measuring the gaps is obtained.

V. Gap Opening Mode

In order to verify the gap opening mode program discussed in chapter 2, another test was designed. In the test which was similar to the previous section, a speckled plate was attached to a caliper, but moving only in *Z* direction as illustrated in Figure 47 and Figure 48. Using the default coordinate system, the values measured by the software are shown and compared to the original values measured by caliper in Table 13: Distance between the plates.

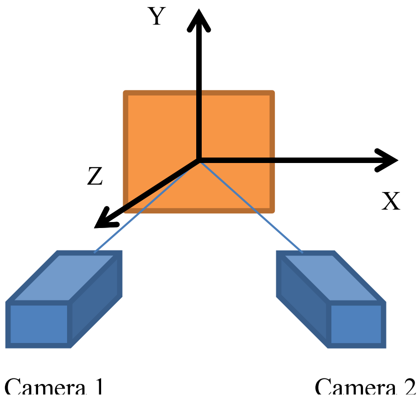


Figure 47: Coordinate system

Table 13: Distance between the plates

Caliper measurement (mm)	Software measurements (mm)	Error (mm)	Error (%)
1.5	1.54605	0.04605	3.069973
2	2.070422	0.070422	3.521121
2.5	2.518156	0.018156	0.726252
3	3.003457	0.003457	0.11522
		Avg: 0.035	

Assuming that the accuracy of the caliper was 0.01 mm, since the maximum error was 0.07 µm, it can be said that the maximum out-of-pane error of the visual system is 60 µm. Likewise, if assuming that the average error of caliper was 0.005, since the average error of the visual system was 0.035, it can be said that average accuracy is 30 µm for out-of-plane movements.

Then a new coordinate system was defined in the gap mode program using the points shown in Figure 48. The increasing distance between point 1 and point 2 was calculated in three directions and shown in Figure 49.

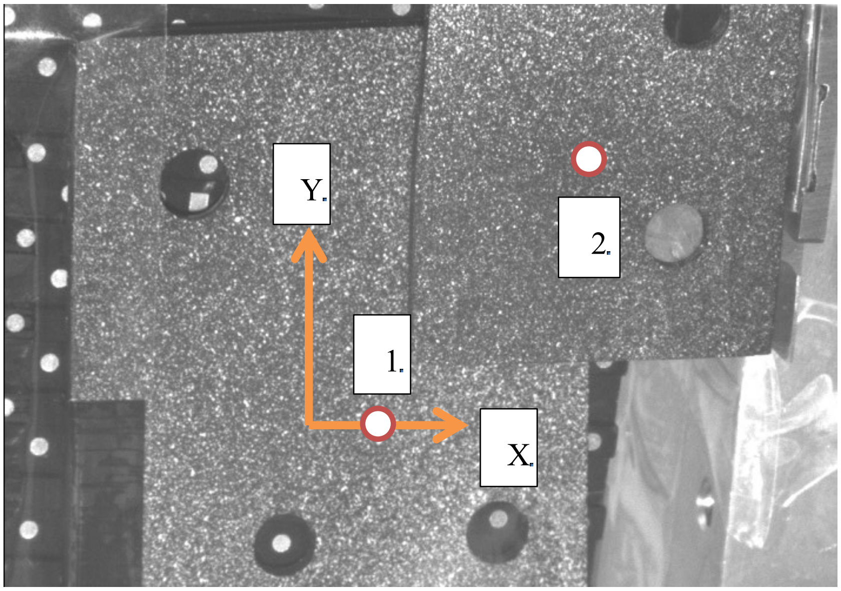


Figure 48: The defined coordinate system and selected points

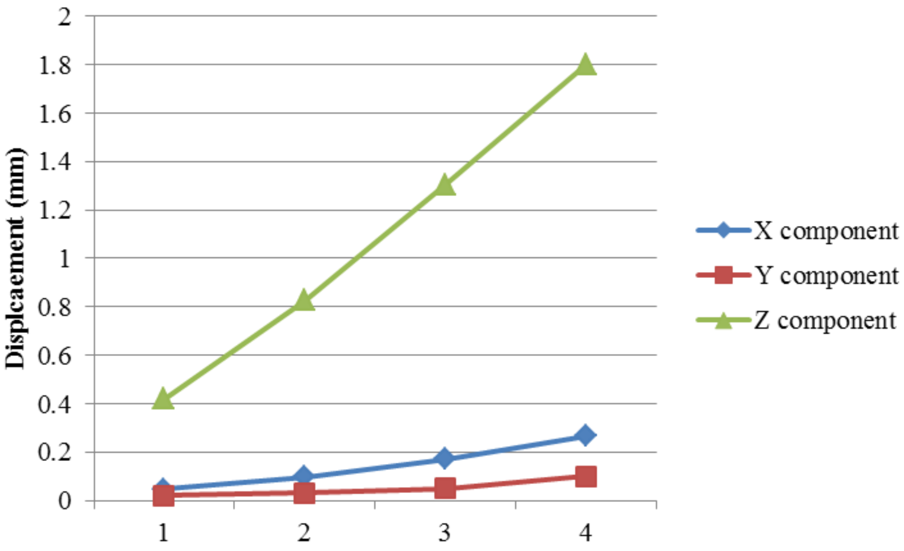


Figure 49: Displacement between point 1 & 2

It is notable that the plate on which the coordinate system was defined had an orientation giving displacement on the other directions too. However, due to being approximately perpendicular to the movement, the major movement component is in still in *Z* direction. This validation example primarily shows a mode III crack opening. This reason justifies displacement less than measured in the default coordinate system of the software. Nonetheless, the main purpose of the program which was detecting the displacement component was confirmed. The compound behavior of crack opening of a pelvic fracture will be shown in Chapter 4.

VI. Repeatability of Strain Measurement

To evaluate the strain measurement, two steps should be taken. In the first step, the precision or the repeatability of the method should be evaluated. In the second step, the accuracy of the obtained strain results should have been assessed. This section looks into the designed test to evaluate the repeatability of strain measurement on a piece of rubber.

Since a material with large deformation capability was required, a rubber part was a good candidate for the test. An abrasion-resistant natural gum rubber sheet with the thickness of a half inch had been purchased. A piece with dimensions of 1” × 6” from a rubber sheet (0.5” thickness) was used as the test piece. The surface of the rubber was covered with a mixture of acrylic polymer emulsion paint and seeding particles with an airbrush gun to create the speckled pattern which can be seen in Figure 50.

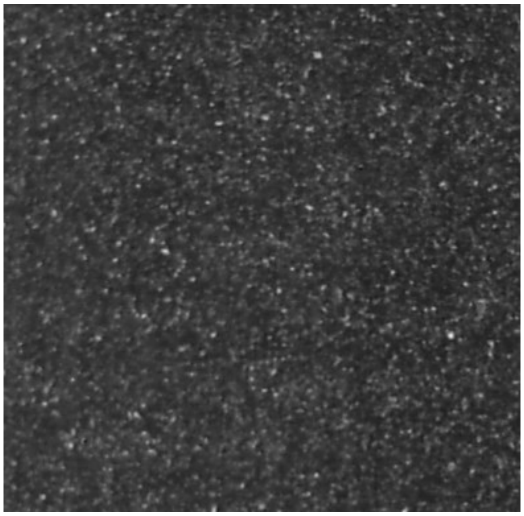


Figure 50: Seeding particle pattern covering the specimen

A notch with the length of 1/3 inch and depth of ¼ inch was made on the sample to have a non-uniform strain field over the surface. (Figure 51)

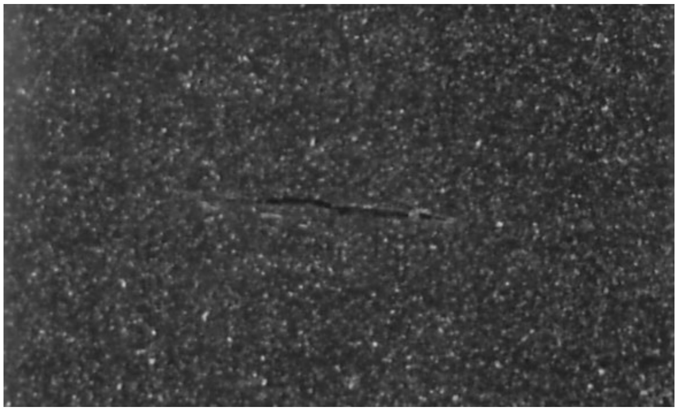


Figure 51: Notch made in the specimen

The test piece was hooked to MABTA using two C-clamps. After clamping, the effective length of the rubber reduced to 4.5”. A Microsoft webcam was used for the test. The test was repeated twice to have three sets of images. All images were taken at the same movement and force which is listed in the Table 14. These steps were: 0.050”, 0.10”, 0.150”, 0.200”, 0.0250”, 0.300”, and 0.350” which were measured with a dial indicator having the precision of 0.001”. The error of repeating these movements was the maximum of 0.002”. This error in different tests occurred because of the difficulty of stopping the machine at the same height level. The error of the load cell was previously found to be around 2.5 N as it oscillated.

Table 14: Displacement and force for the first three sets

Displacement (in)	Displacement (mm)	Force (N)			AVG (N)
0	0	51	50	52	**51**
0.050	1.27	65	65	65	**65**
0.100	2.54	82	82	82	**82**
0.150	3.81	96	94	94	**94.7**
0.200	5.08	108	104	105	**105. 7**
0.250	6.35	116	114	114	**114. 7**
0.300	7.62	128	125	128	**127**
0.350	8.89	139	128	136	**134. 3**

The data were then processed using a commercial image processor software (DaVis 8, LaVision GmbH). The images were processed using a square subset 65 pixels X 65 pixels, and 8 pixels distance between calculated points. Figure 52 shows the deformation field of the part. That top of the piece moves upward while the lower areas have downward movement. Also, due to the poisson ratio, even though there is only vertical loading on the part, it contracts in the transverse direction of the loading.

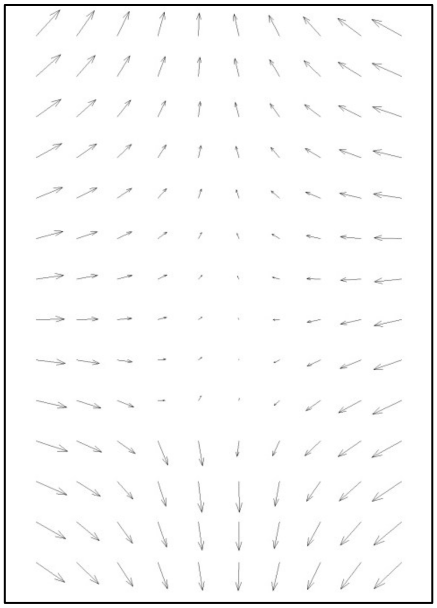
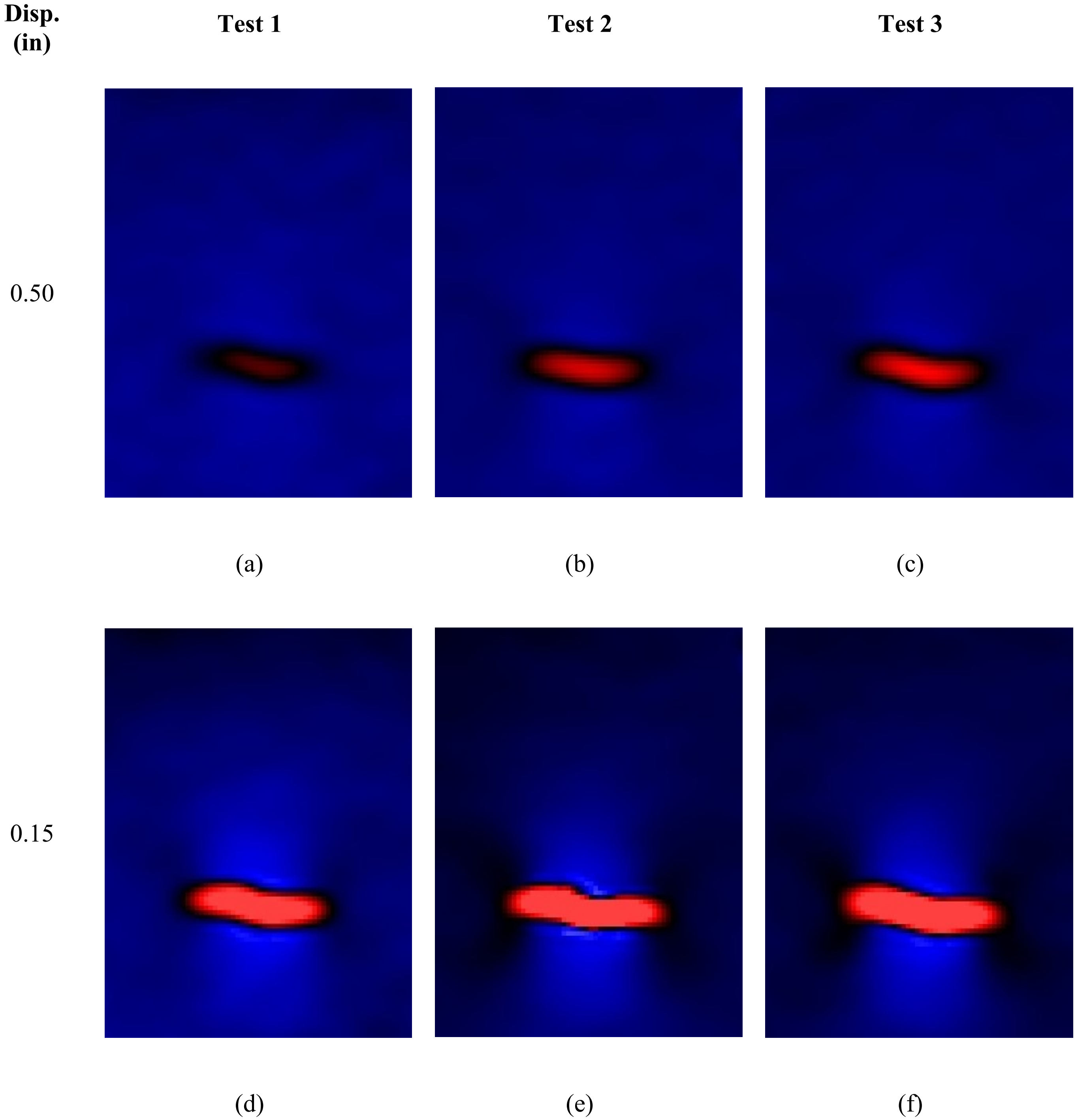


Figure 52: Displacement field obtained after image processing

Figure 53 shows the strain field during the experiment for the tests 1, 2, and 3. The color map used in the strain fields can be found in Figure 54. In Figure 53, the areas in the vicinity of the crack tip are subjected to higher amount of strain, whereas the top and bottom of the crack, where no strain flow is anticipated, show significantly lower strain. This figure can be used for qualitative comparison between the tests. The color of the strain field was chosen as can be seen in Figure 54. For simplicity, only the even increments corresponding to 0.05”, 0.15”, 0.25”, and 0.35” are shown.



(a) (b) (c)

(d) (e) (f)

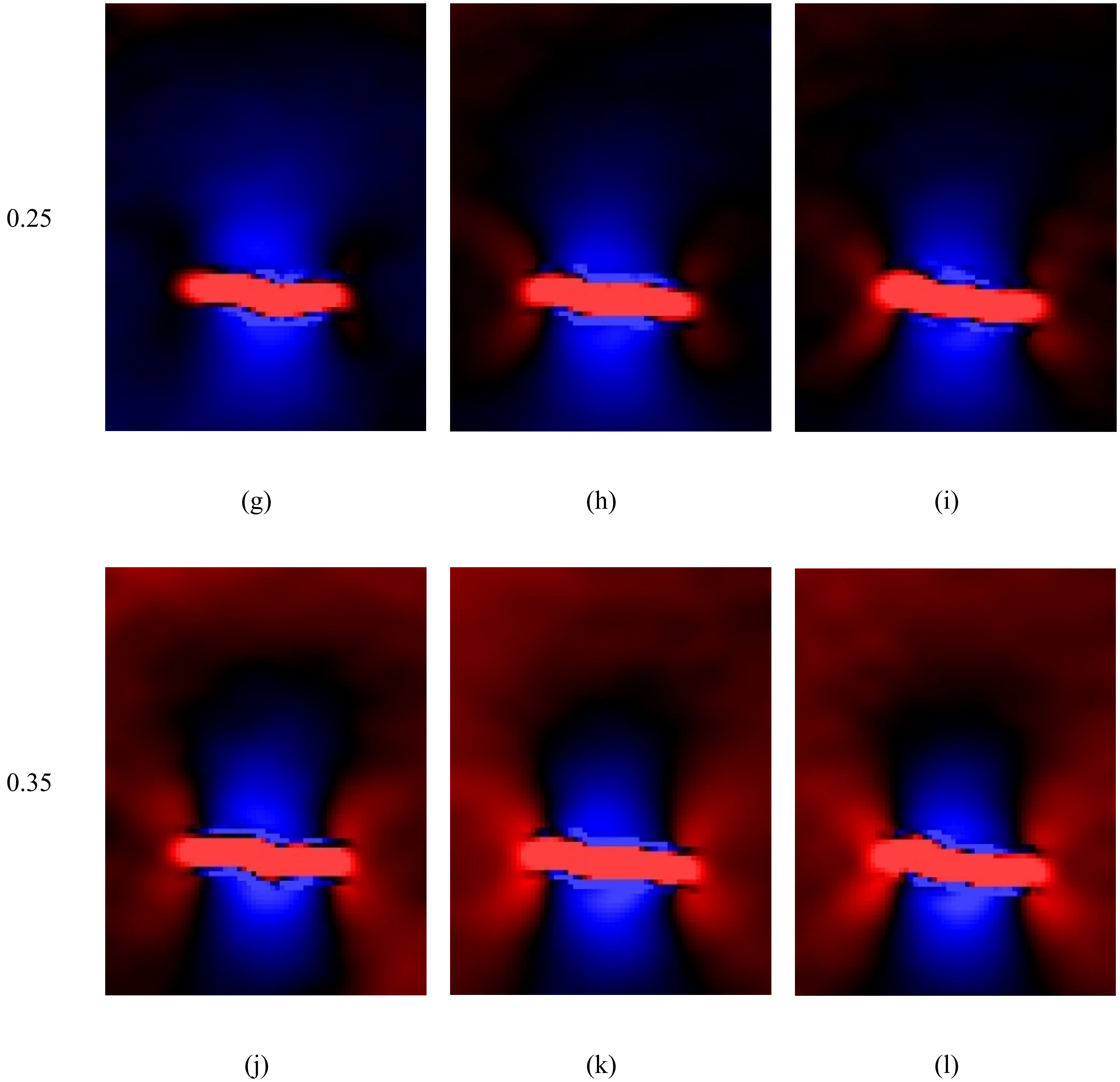


Figure 53: Strain field of the rubber sample. Each row shows the field for increasing overall displacement of 0.1" from 0.05" to 0.35". Each column shows a set of images corresponding to different tests to evaluate repeatability.

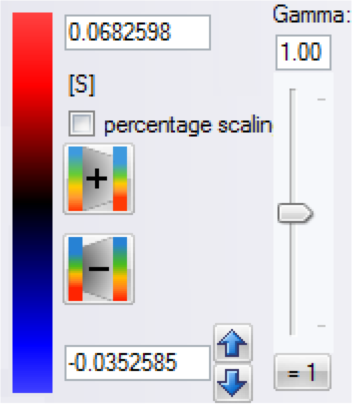


Figure 54: Color map used for strain fields

The tests 2 and 3 are more similar than the test 1 although all of them are at the same range similar enough to see the repeatability. The difference in the test1 is that it seems to have a lower strain than the rest. It is possible that during the experiment, the notch had been growing, resulting in more severe strain and stress gradually.

It is essential to have a numerical comparison to find out if this error is small enough to be ignored. For evaluating the numeric magnitude of an arbitrary point, two points were chosen in negative and positive strain areas. However, due to the limitation of the software, an area must have been chosen to calculate and plot the strain value rather than a single point. Two areas were chosen seen in Figure 55: above the notch (area A), where minimum strain can be seen, and near the notch tip, but a bit upper, where the maximum strain occurs (area B).

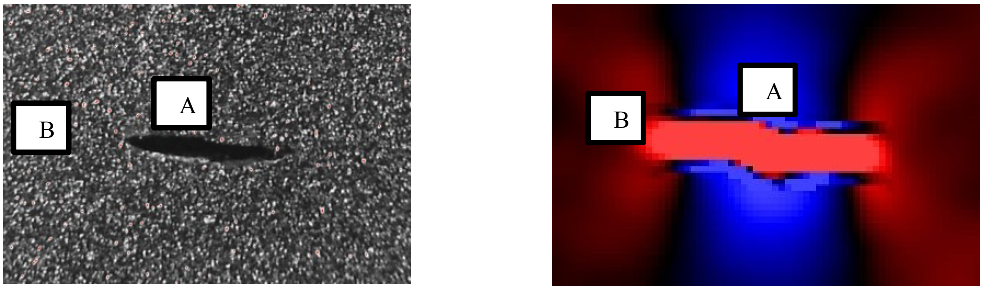


Figure 55: Chosen area to calculate and compare the strain magnitude

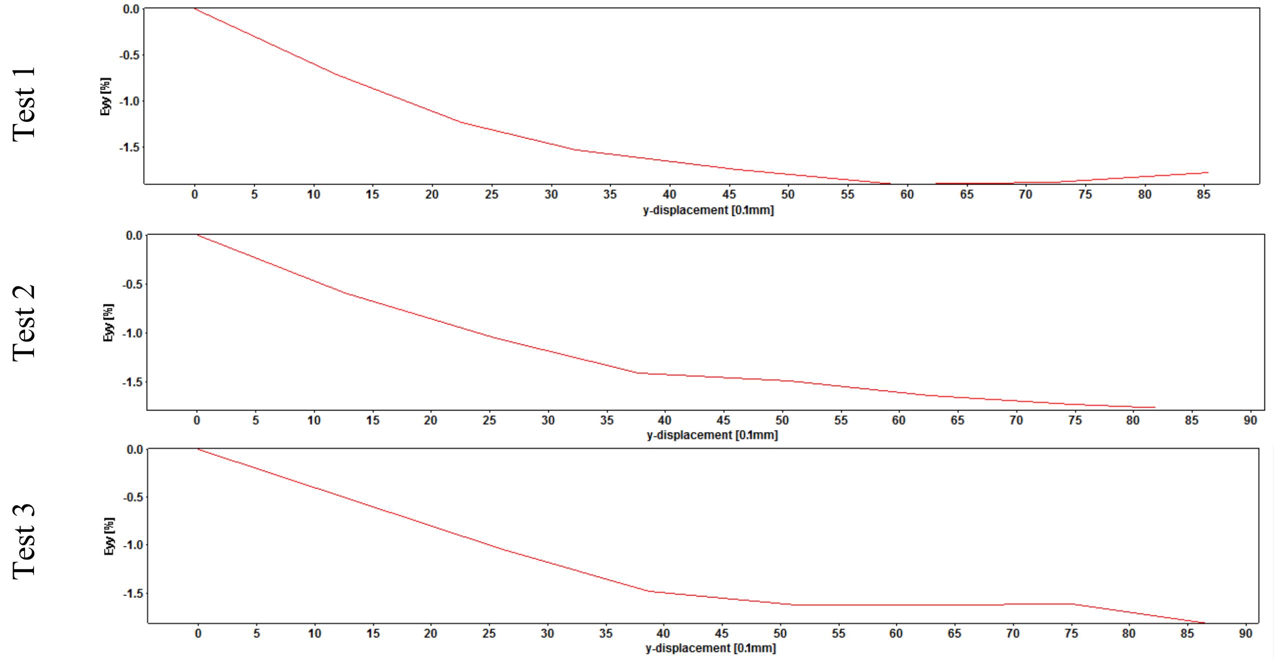


Figure 56: Strain magnitude for area A

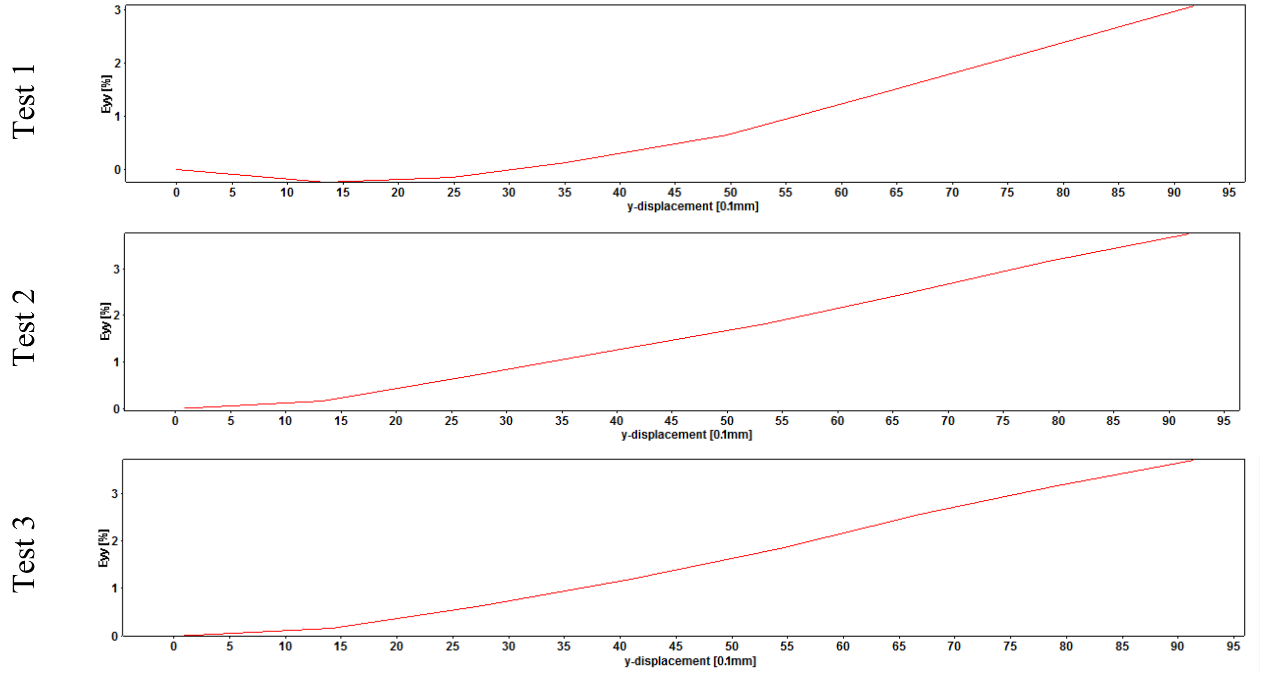
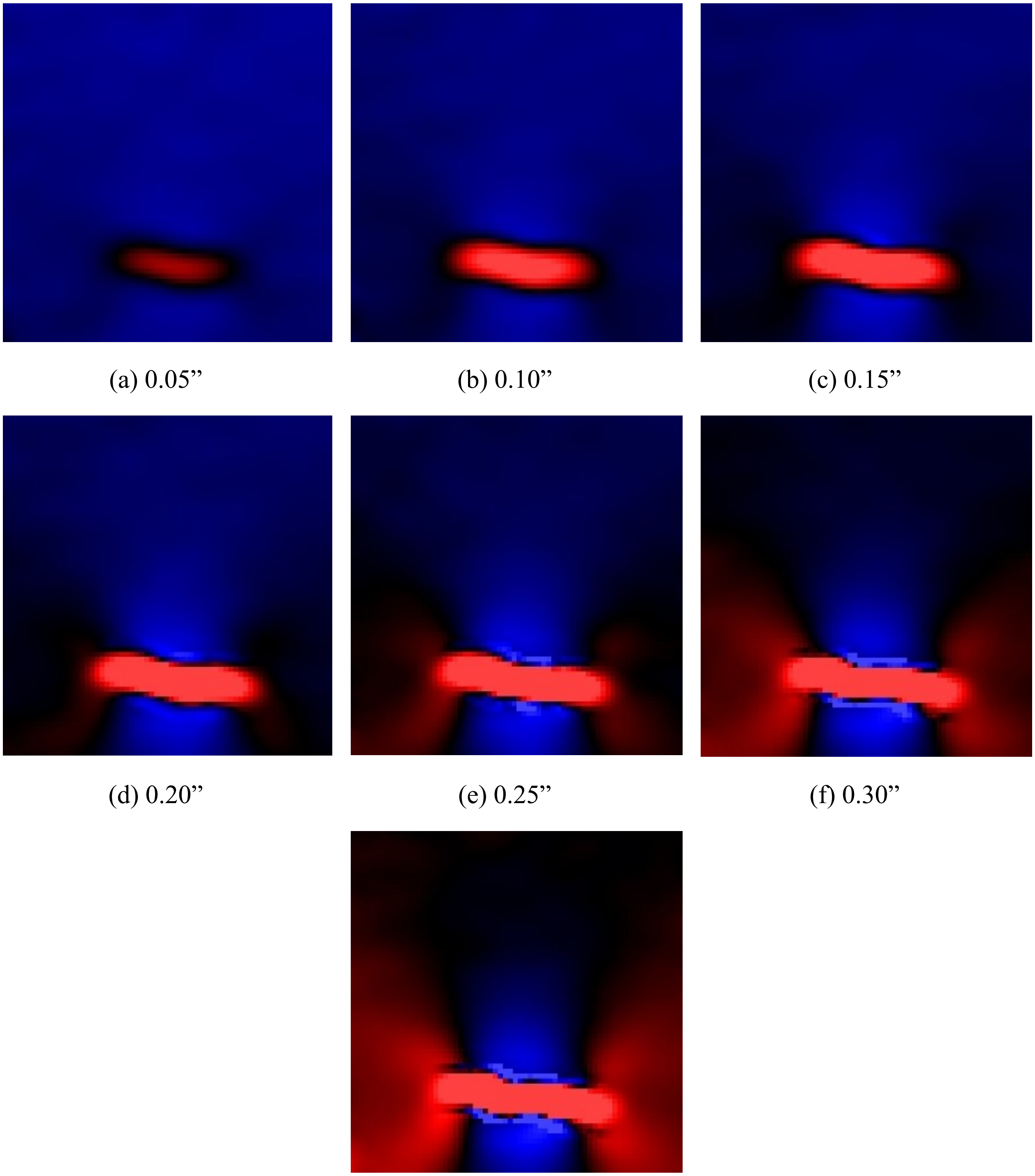


Figure 57: Strain magnitude for area B

Figure 56 and Figure 57 agree with possibility of notch growing since both positive and negative strains have smaller magnitude at the end. The only strange behavior can be seen in the test 1 of Figure 56. The reason that the strain dropped in around 6 mm and curved up is not clear. It can be assumed that at that point the notch started growing. However, Figure 57 does not show this behavior at 6mm, but there another strain drop in 1.5 mm. The possible explanation for these behaviors in the first test is that the notch could be growing in such a way that the strain dropped at the area B first and after higher loading, the notch grew in a way that the decreasing strain could be observed in the area A. It should be noted that the notch was not through all, and it could grow laterally toward the sides as well as more in depth. So the reason behind the odd behaviors on the first test could be explained by changing the geometry of the notch in the first test and remaining constant in the others. The other tests show similarity.

The setup was dismantled and assembled again to have the 4th the test. In the last test, the results are expected to be more different than previous results since the test piece was taken off and on. The reason is that potentially the effective length of the test piece and/or the positions of the grips could have changed. The strain field with the same color map as before can be found in Figure 58. In terms of numerical comparison, Figure 59 and Figure 60 show the strain magnitudes in the areas A and B.



(a) 0.05”

(b) 0.10”

(c) 0.15”

(d) 0.20”

(e) 0.25”

(f) 0.30”

(g) 0.35”

Figure 58: Strain field corresponding to test 4. Each image shows the strain field after 0.5” increasing in the total displacement.

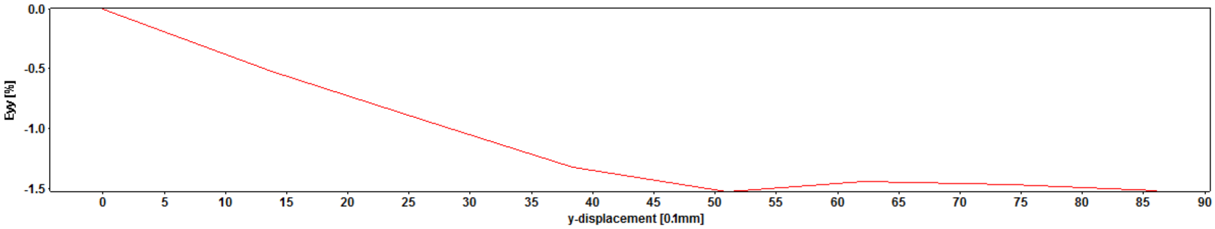


Figure 59: Strain magnitude for the area A in test 4

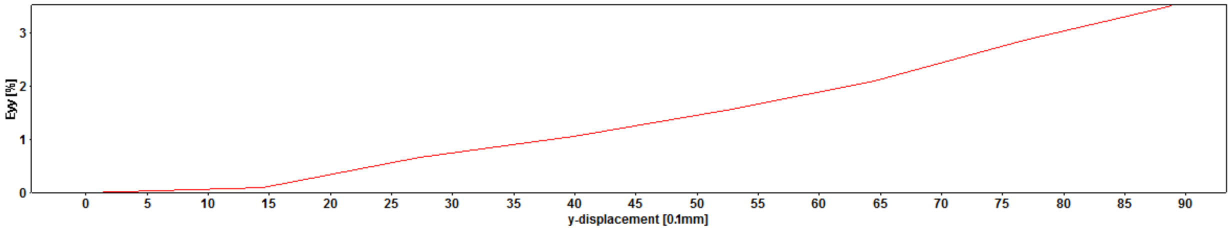


Figure 60: Strain magnitude for the area B in test 4

The magnitude of strain in positive region is the same as the tests 2 &3. The only issue is the repeated strange behavior in the area A. However, all numbers are close comparing the final strain at the points.

For the region B, the magnitude in the tests 2, 3 and 4 are approximately equal. The trends in all tests are also similar. In the region A, it should be noted that this region was chosen very close to the notch and the strange behavior in test 1 & 4 could be because of this sensitive area. The proximity to the boundaries could result in higher error than the other points.

In this section the repeatability of the test was assessed which was shown on the entire field, and also it was assessed quantitatively on two extreme points. In the next section values obtained in the strain measurements are evaluated.

VII. Evaluation of Strain Measurement

On the previous step the repeatability of strain measurements was confirmed; however, the obtained magnitudes should have been evaluated. For validating the strain values obtained from the DIC setup, another test was designed. The test was done on dog bone test pieces made of polypropylene with the known strain stress behavior. The samples were made by injection molding and not annealed. Therefore, residual stress could have been found in the parts; however, the previous tests on the similar parts, done by another student, had showed that the behavior was expected to be quite linear in the elastic area. Since the samples were white as well as the particles that were available for speckling pattern, they needed to be painted black. A flat black paint was sprayed on the parts. With same technique that had been used for the previous experiments, a speckling pattern was made on the test pieces. Two samples were made and prepared to run the DIC on. For the tensile test, two grips were made to hold the samples. Having pulled the samples, the displacement of the grip and the vertical force were read from the MABTA program. A pair of pictures was taken from the initial position and every step.

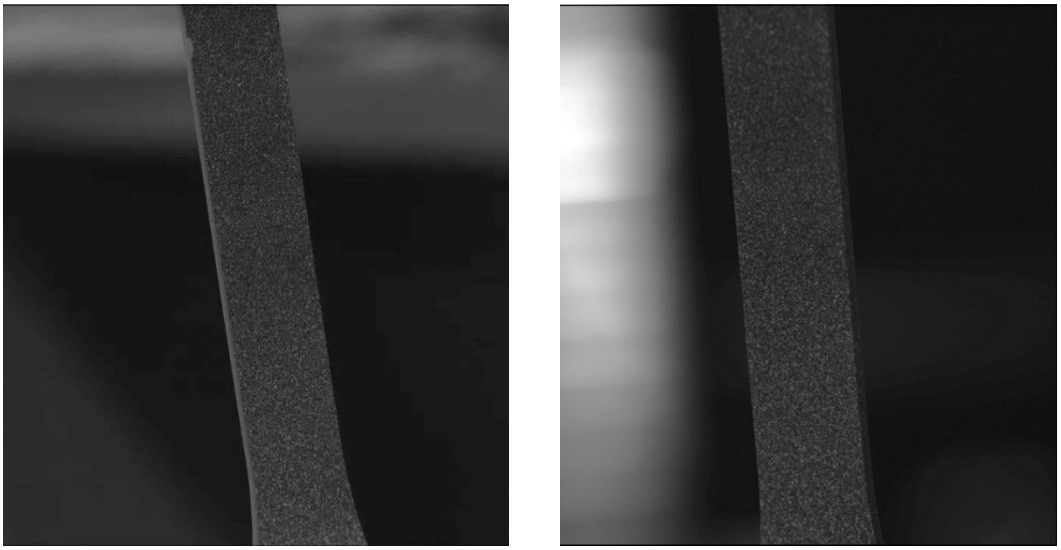


Figure 61: Raw images taken from the part.

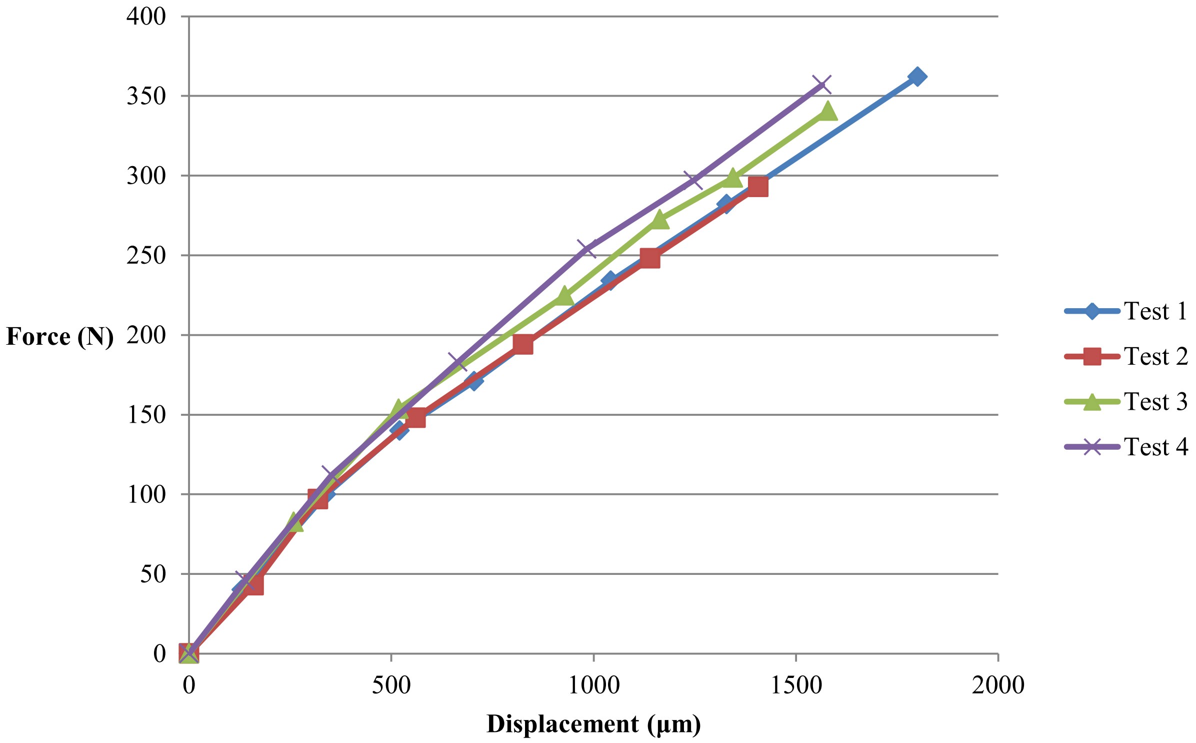


Figure 62: Measured vertical force vs the displacement of the lower jig during the tests

After processing the pictures in the commercial image processor software (DaVis 8, LaVision GmbH), the normal strain in the longitudinal direction was obtained using the subset size of 65x65 pixels. A total of 6 tests were initially done. However, after the processing, a non-uniform strain distribution was found in the parts, suggesting that one side of the part was under more tension than the other side. Investing the set up, it was found that the lower jig was holding the pot at an angle resulting in an uneven tension. Therefore 2 tests were assumed to be erroneous and removed from the figures leaving 4 sound tests to be investigated. The strain field of one of the erroneous tests can be seen in the Figure 63. In Figure 64 the strain results of the test 3 can be seen.

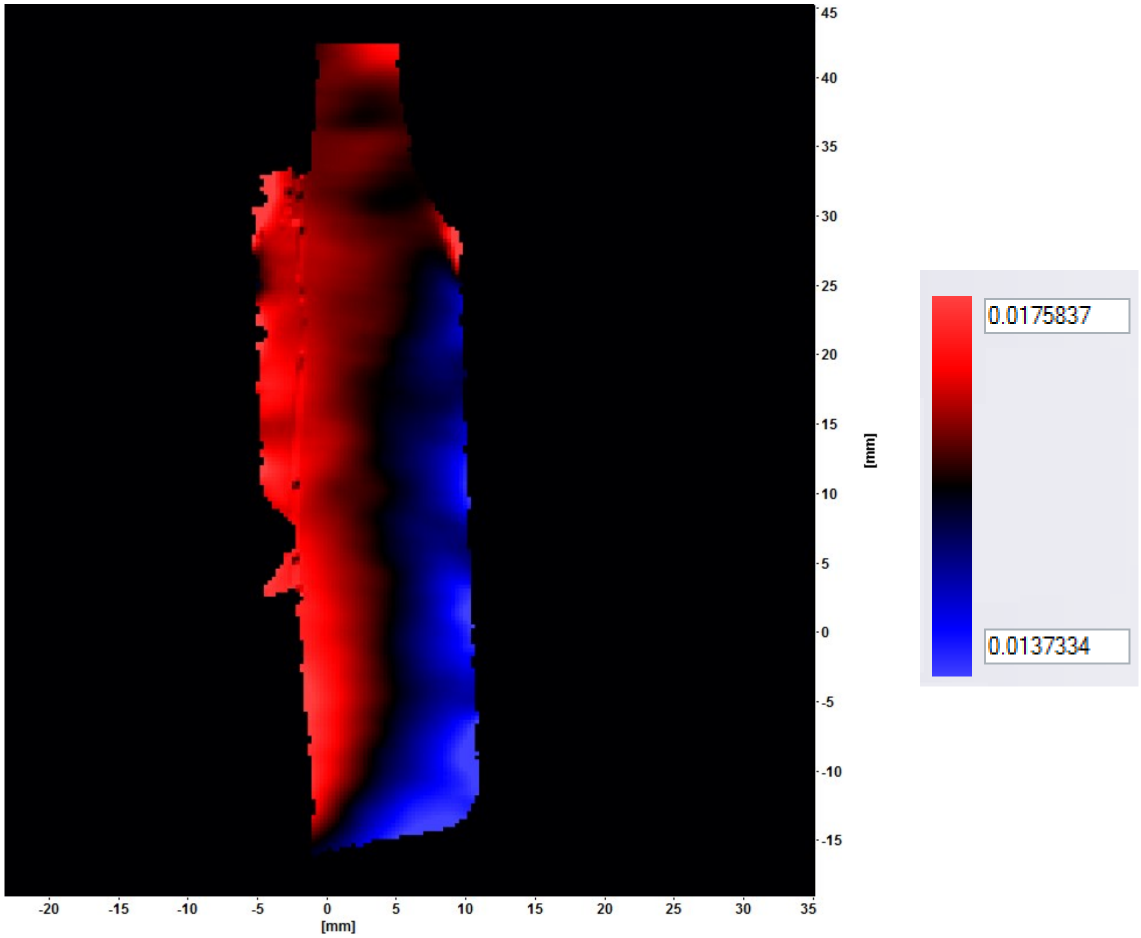
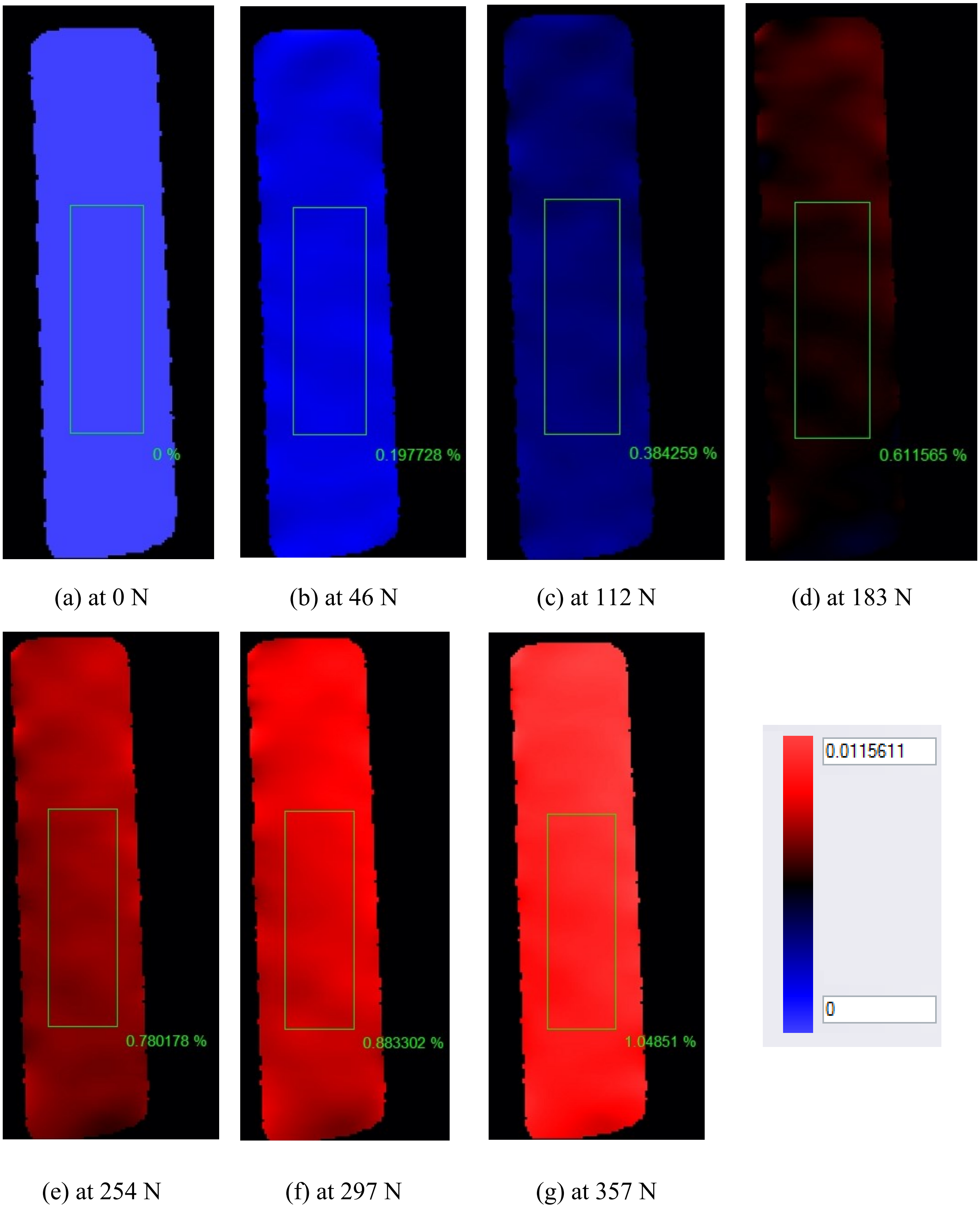


Figure 63: Strain field of a test with uneven tension. The results of the test were pulled out. It can be seen that the left side of the sample was pulled more than the right side was.



(a) at 0 N

(b) at 46 N

(c) at 112 N

(d) at 183 N

(e) at 254 N

(f) at 297 N

(g) at 357 N

Figure 64: Strain field distribution corresponding to the test #3. The used color map can be seen on the right, and the corresponding forces can be seen below images.

The other issue that was investigated was the distribution of the strain on a single state. In Figure 65, the strain distributions of test #2 and #3 which were one on two different samples can be seen. The color maps were chosen to show the maximum and minimum strain found in the part. The first point to mention is that the measured strain is in a very narrow range, and the other is the ability of DIC to observe the non-uniform accurately. It can be suggested that the residual stress and material imperfection on the first sample led to having a very non-uniform strain distribution while this matter is less seen in the second sample. As mentioned earlier, the sample were not annealed which can explain the existence of residual stress in the parts.

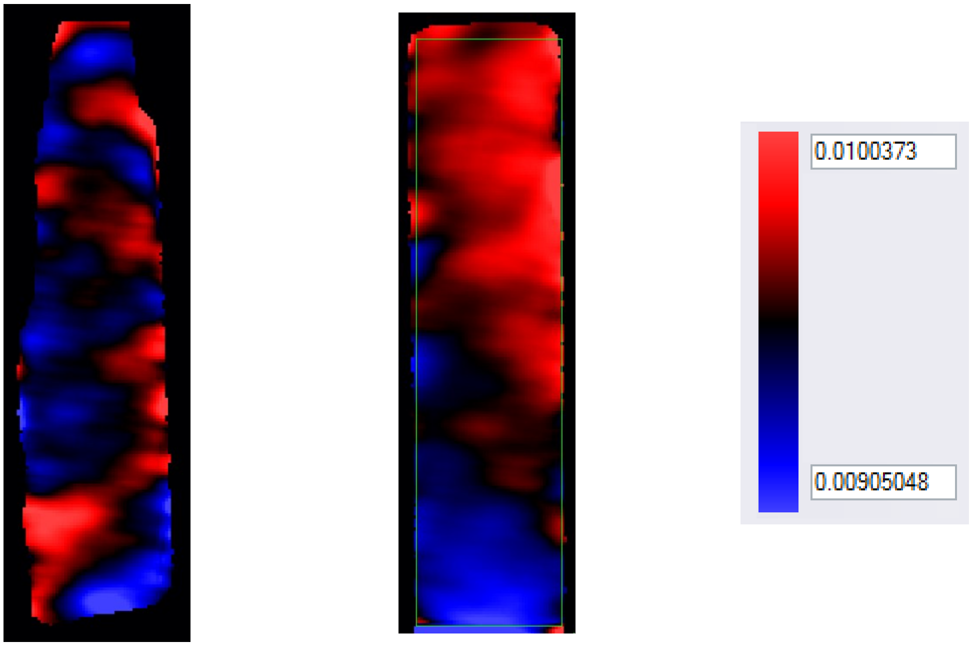


Figure 65: Strain distribution on test #2 & test #3

Comparing the average strain over a rectangle in the middle of the samples (as can be seen in Figure 65) with the analytical true strain, Figure 66 was obtained. To calculate the true strain the formula of $\varepsilon = \ln(\frac{L_2}{L_1})$ was used with the initial length of 135 mm. It is notable that using the force-displacement behavior, it was seen that the material behavior was still in the elastic area which allows assuming simple calculation of true strain. Based on the data, the maximum deviation from the theoretical strain was 11% and the average error was obtained as 6%.

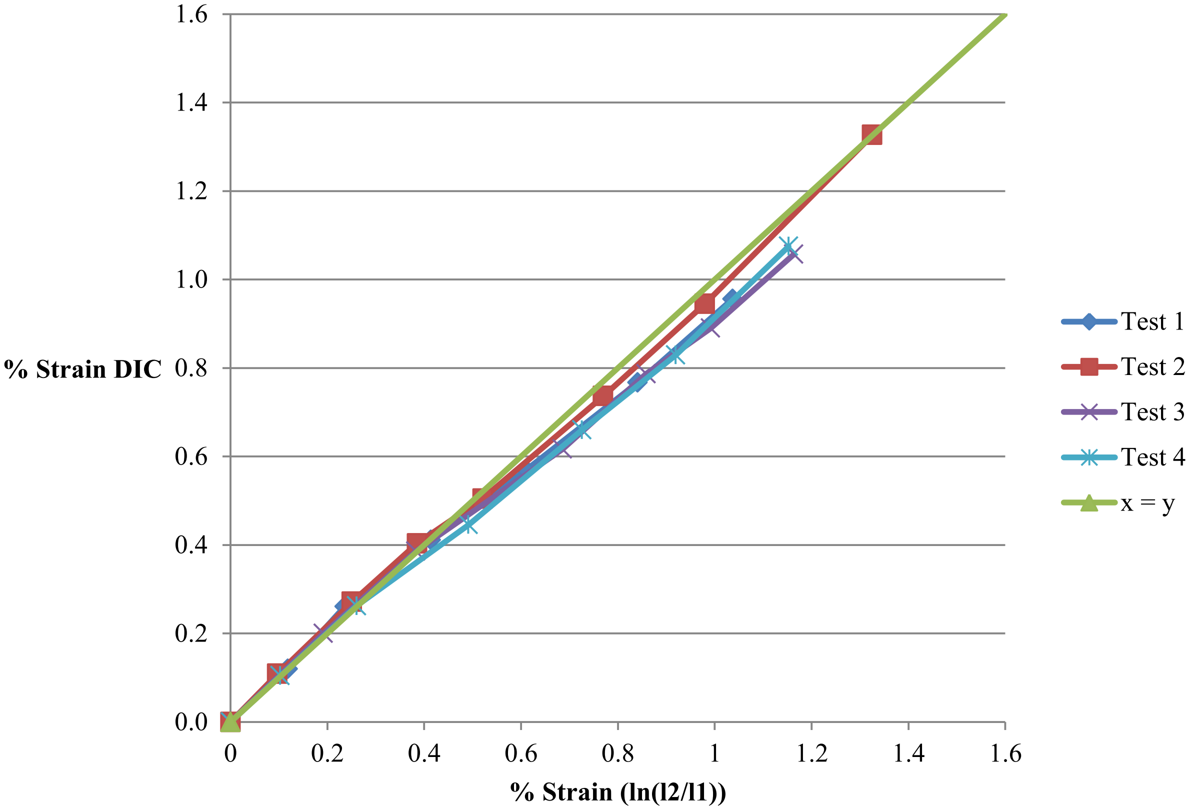


Figure 66: Measured strain vs calculated strain for the test pieces.

VIII. Summary

In this chapter all the setup components were evaluated. The load cell has the accuracy of 2.5 N and the error of 1%. The motor works with the error of less than 1%. The camera set-up detected movements with the error of less than 3.5% or 250µm. The developed Matlab program was implemented to measure the gap opening as well as its mode of opening. The accuracy of measuring the in-plane gap opening was 5µm and the accuracy of measuring the out-of-plane opening was 30 µm. At the end, both the repeatability and the accuracy of the strain measurement system were assessed where the system showed promising repeatability with the average error of 6% in measuring the strain.

Now that the system has been validated it is possible to look at the loads, displacement, strain field and crack opening of a more complicated geometry. The next chapter will look at tests on a hemi-pelvis.

Chapter 4: Methodology and Results

I. Intact Pelvis Test

A left composite hemi-pelvis (fourth generation) was acquired from Pacific Research Laboratories, Inc. (also known as Sawbones). The hemi-pelvis was first painted black in order to have good contrast with the white speckle pattern. Following the previously described method, it was potted. With the same technique used in section 3.VI, a speckled pattern was created. Figure 67 shows the speckled potted hemi-pelvis. Then the set was assembled in the MABTA (Figure 68).

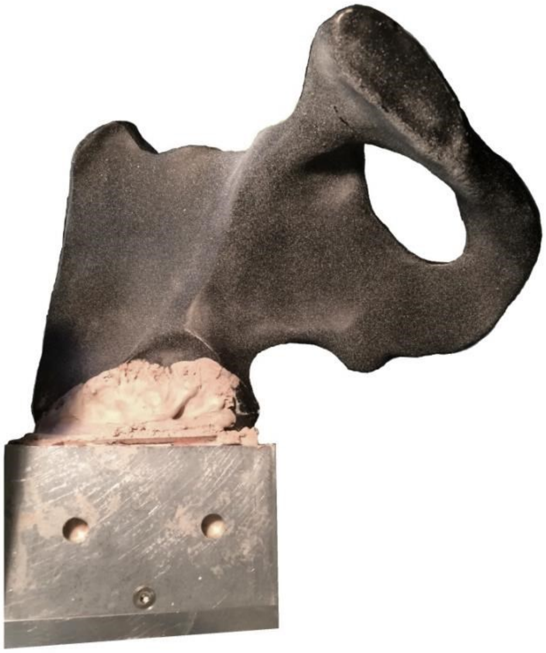


Figure 67: Speckled potted hemi-pelvis

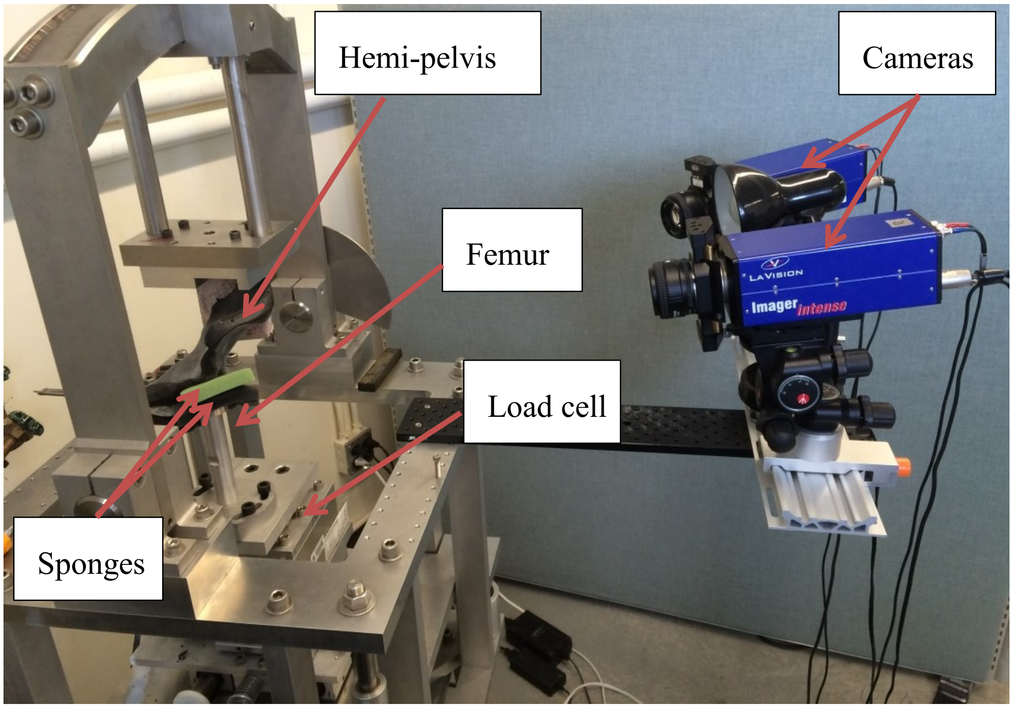


Figure 68: The final set-up assembly

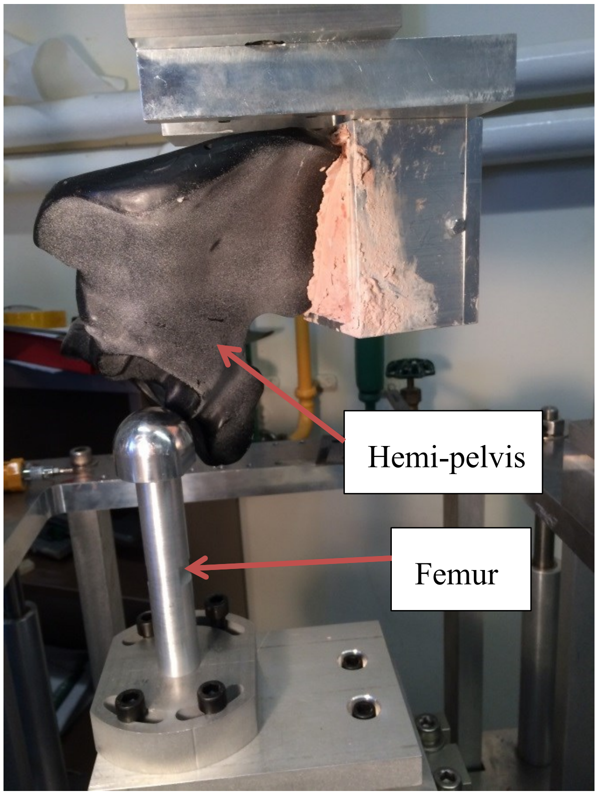


Figure 69: The view that cameras had of the hemi-pelvis

When assembling the potted hemi-pelvis into the MABTA, the prosthetic femur could not be aligned with the hip cup. Therefore, another component was manufactured by which the femur prosthesis was aligned with the hip joint. The other concern was the possibility of squishing and crushing the inner part of the acetabulum. The reason was that the acetabulum inner surface was not perfectly matching the spherical femur prosthetic shape. With the absence of the cartilage tissue between the two parts, it was possible that the femur would create high stress concentrations on the acetabulum. Therefore, some sponges were used to replace the cartilage tissue preventing any damage to the composite bone, although they might not behave exactly same.

Then the region of interest (ROI) on the part needed to be determined. Using the simulation done on the pelvis in chapter 2, and other studies [39] the area of interest was chosen so the area with high strain could be observed. Not only are higher strains expected, but the location of insufficiency fractures is commonly found in the oblique iliac region shown in Figure 70 [12].

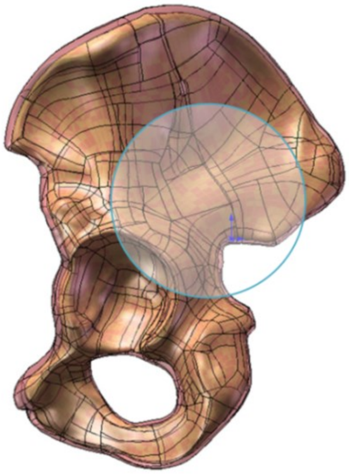


Figure 70: Region of interest on hemi pelvis

Then the cameras were set up so they can have a good view of the region. For a better contrast a lamp was used. The position of the cameras can be seen in Figure 71. It can be seen that the angle between the cameras is 25 degrees (rather than the usual 30-45 degrees, which was used up to now) and is slightly smaller than the suggested range in other studies (between 30 and 90 degrees) [77]. However, changing the angle at this range would not add a significant error [77]. The reason of having this angle was that it was the largest angle by which it was physically possible to look at the ROI.

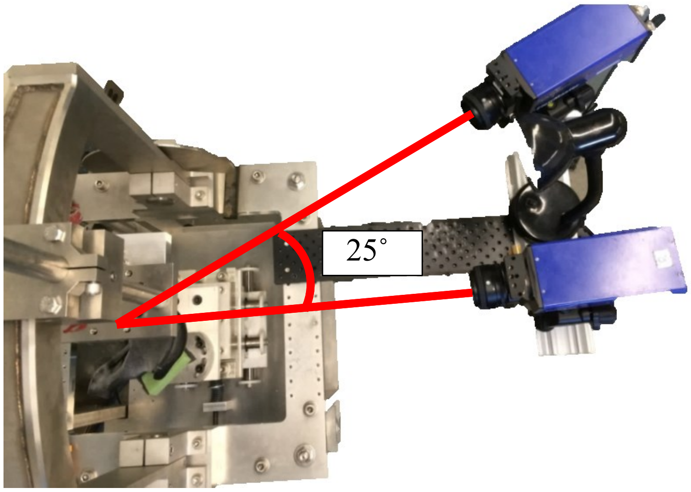


Figure 71: Cameras positions showing the angle between them

The cameras were calibrated and the set up was ready for testing. During preliminary tests, it was observed that after movement of the femur, the measured force dropped. This observation was expected since bone has viscoelastic properties [78]–[81] and the sawbones models are similar to actual bones. Figure 72 shows that after one minute relaxation, the loading reaches a plateau. Based on this observation, all loadings were recorded one minute after displacement.

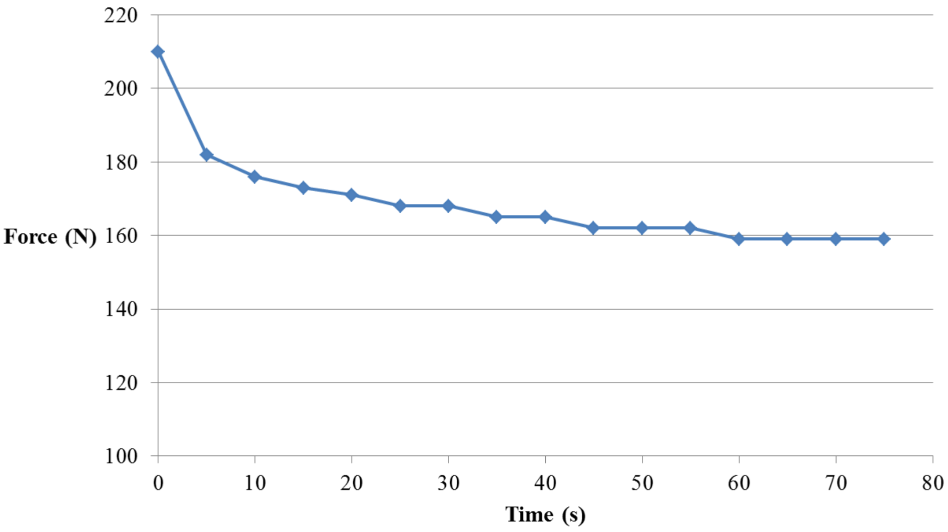


Figure 72: Relaxation during 75 seconds following a step loading

For testing the hemi-pelvis, the femur prosthetic was displaced until a vertical loading of ~50N was reached. The femur position was adjusted during one minute relaxation to have a constant loading of 50N. Then a pair of pictures was taken of the region of interest. This procedure was repeated again with the increments of ~50N until the final loading of 600N. Figure 73 shows the loading displacement of the test piece. It shows a linear behavior after loading of 100N.

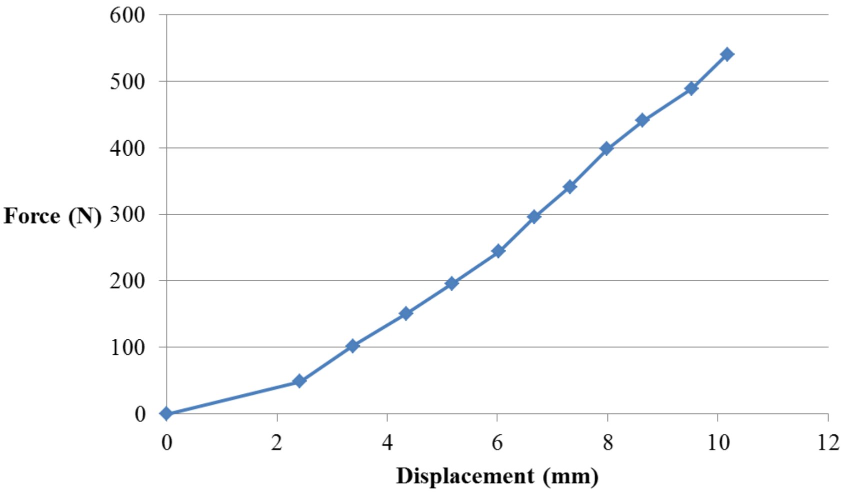
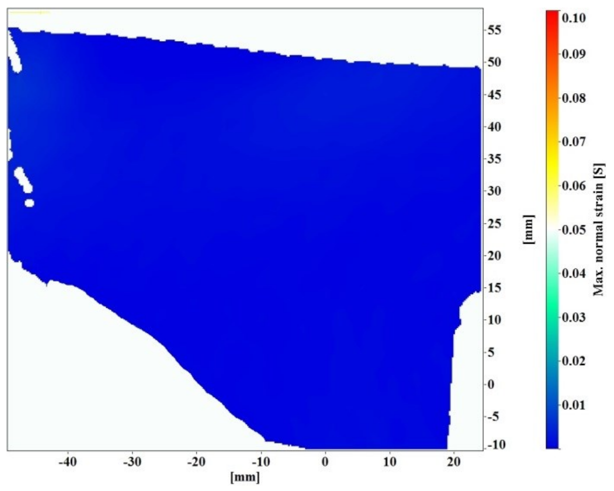
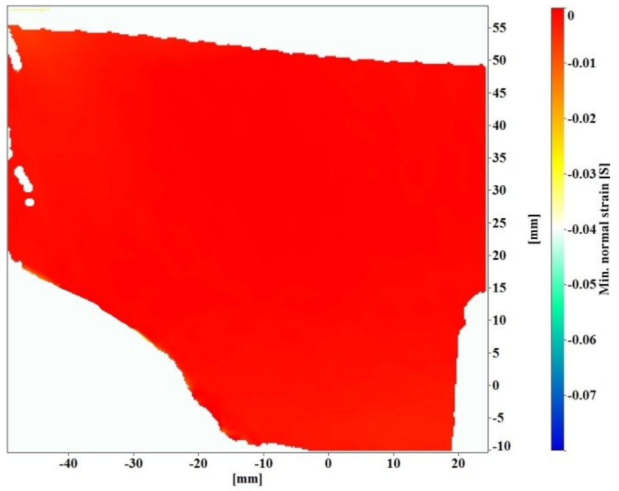


Figure 73: Force-displacement curve obtained from testing on the healthy pelvis

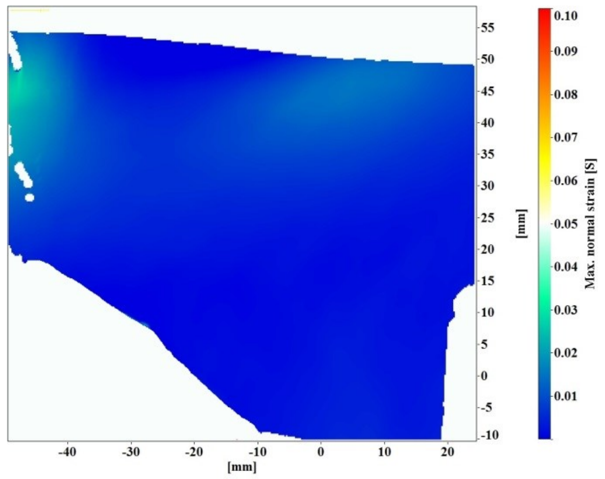
Images were processed using the commercial software (DaVis) to investigate the strain field. Figure 74 shows the minimum and maximum normal strain field while increasing the loading. For simplicity, only the even increments are shown, meaning that the forces corresponding to the presented fields are 100N, 200N, …, 600N. Since the color map is kept constant to enable the reader to compare the figures, rising strain is not easily recognizable in the first few images. Figure 74 shows that the magnitude of maximum normal stain is larger than the minimum normal strain in most parts. Therefore the dominant behavior is tension. Also the area of high strain can be seen at two regions on the top. This predicts that if a crack is going to be generated or propagate, it would be at or towards these areas.



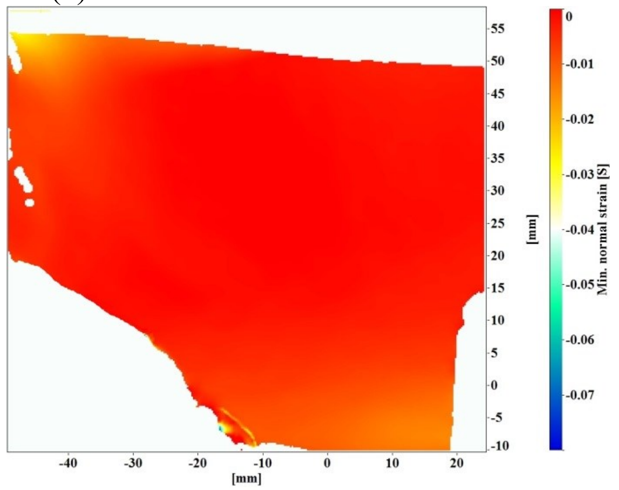
(a) The maximum normal strain at 100N



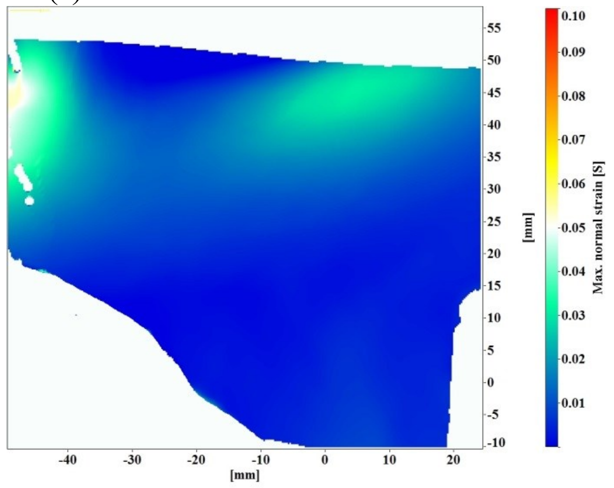
(b) The minimum normal strain at 100N



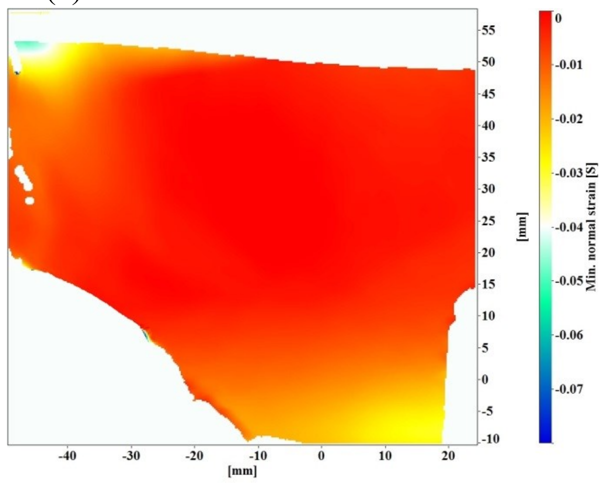
(c) The maximum normal strain at 200N



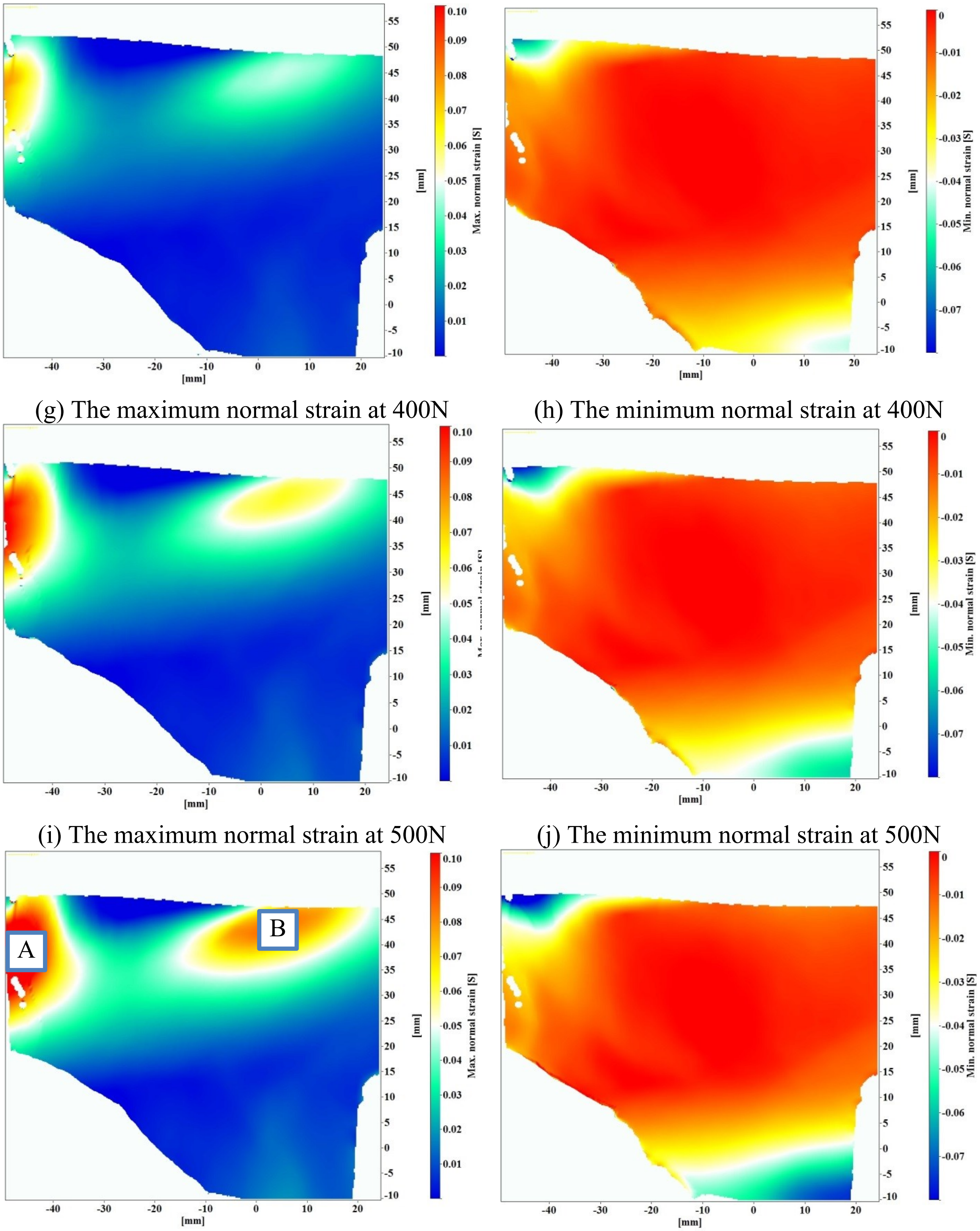
(d) The minimum normal strain at 200N



(e) The maximum normal strain at 300N



(f) The minimum normal strain at 300N



(g) The maximum normal strain at 400N

(h) The minimum normal strain at 400N

(i) The maximum normal strain at 500N

(j) The minimum normal strain at 500N

(k) The maximum normal strain at 600N

(l) The minimum normal strain at 600N

Figure 74: The strain field of the region over subjected to different loading. The left column shows the maximum normal strain for each frame whereas the right column shows the minimum normal strain. Also each row shows a 100 N increase in loading from 100N to 600N.

For quantitative evaluation of the strain on the pelvis two regions were chosen as can be seen in Figure 74.k. The maximum and minimum normal strain during the loading for these two regions can be Figure 75 and Figure 76.

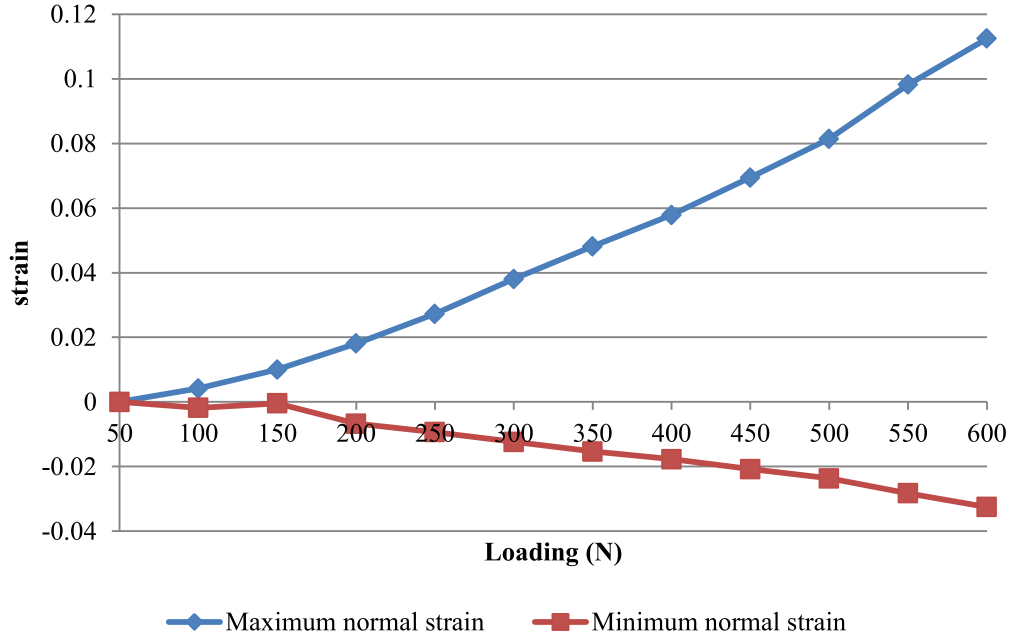


Figure 75: Maximum and minimum normal strain of the region A shown in Figure 74.k

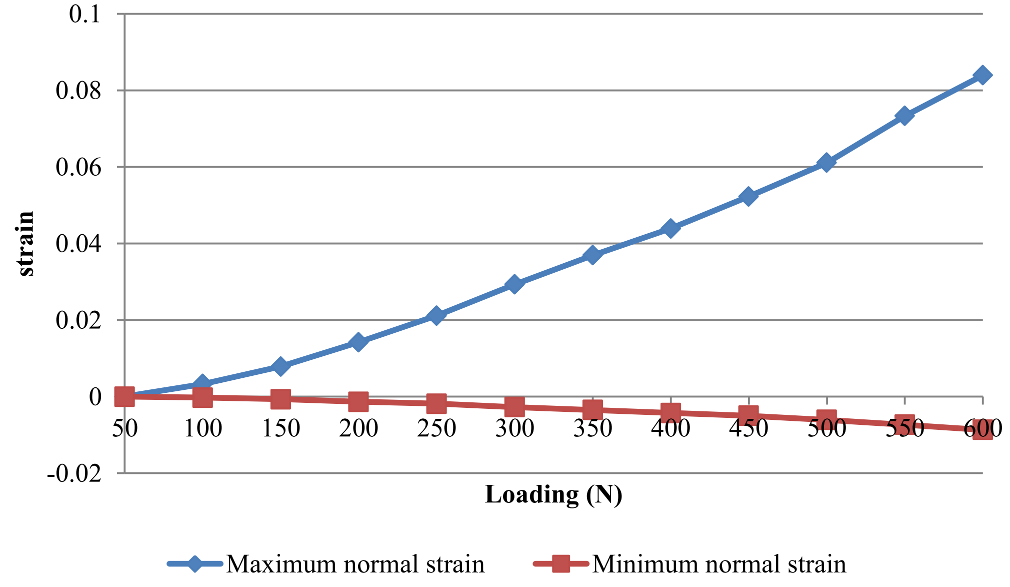
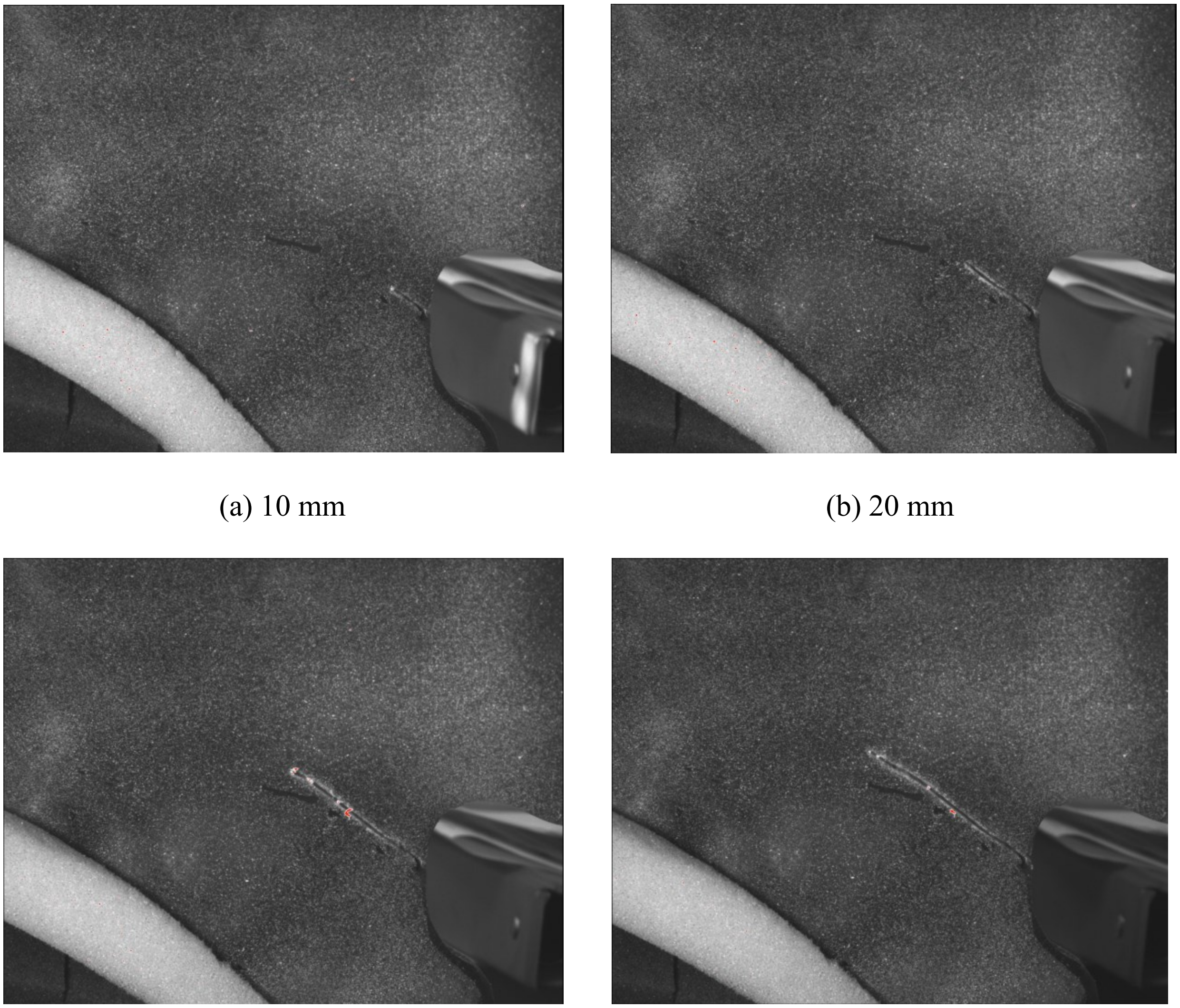


Figure 76: Maximum and minimum normal strain of the region B shown in Figure 74.k

II. Broken Pelvis Test

The objective of the test was to investigate the change in the strain field of a broken pelvis as well as measuring the gap opening. So the pelvis was cut by a saw to have an increasing length of the crack. Its length was increased by 1 cm each time. The increasing crack during the test can be seen in Figure 77.

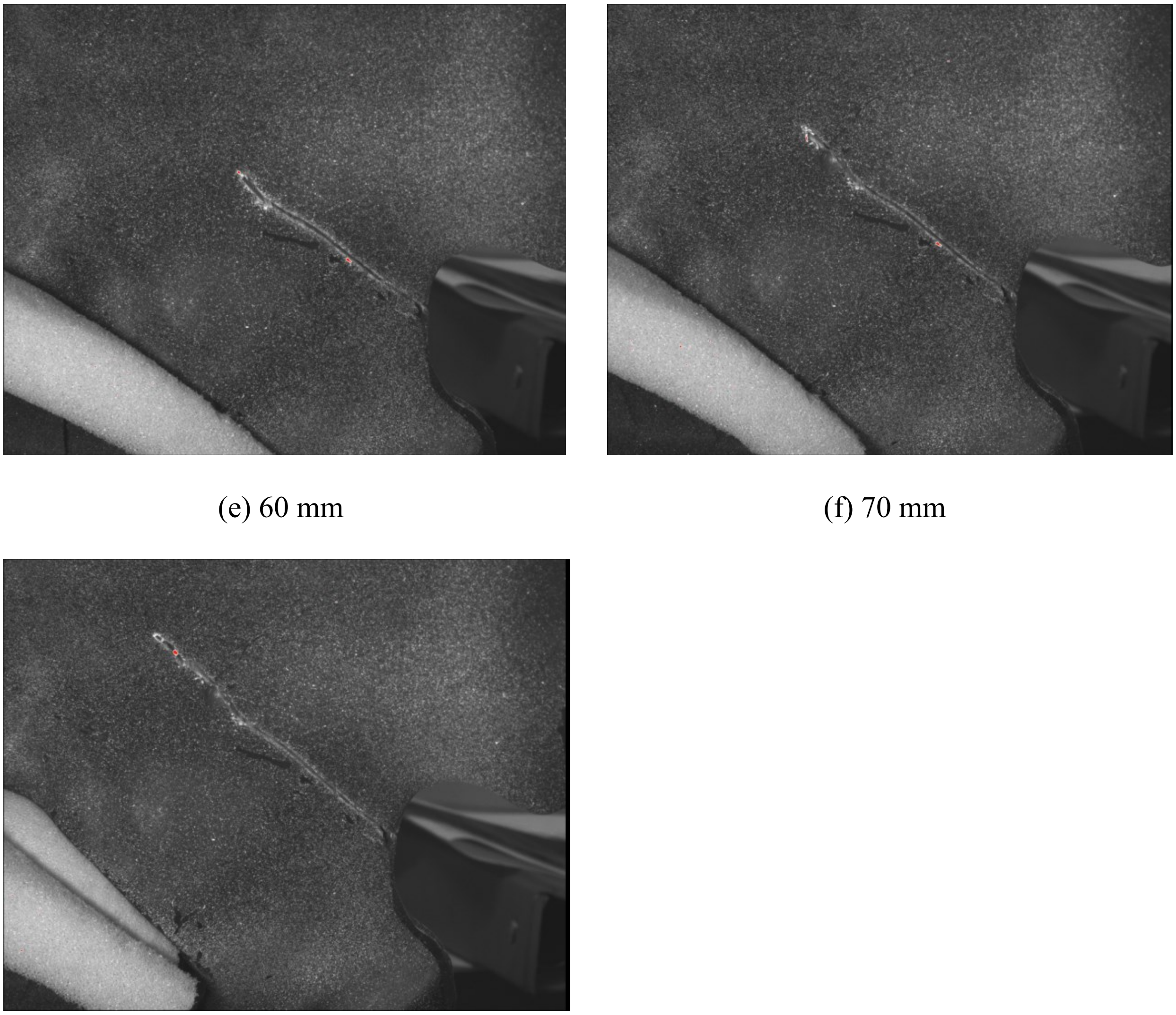


(a) 10 mm

(b) 20 mm

(c) 40 mm

(d) 50 mm



(e) 60 mm

(f) 70 mm

(g) 80 mm

Figure 77: The increasing crack created by a saw on the hemi-pelvis

For the first 4 tests (cut length seen from Figure 77.a to Figure 77.d), the test was repeated again. The loading interval was similar to the intact pelvis test, taking a pair of images at each step. The load-displacement curve corresponding to cut length 10 mm to 50 mm can be seen in Figure 78a-d.

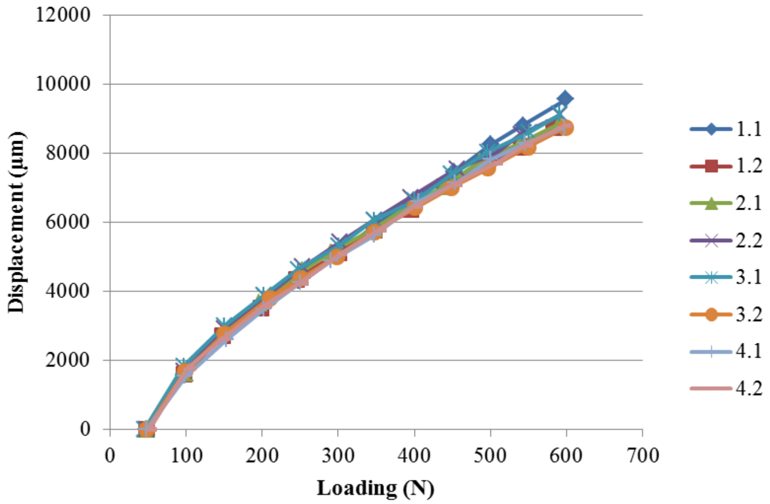


Figure 78: Displacement vs loading for the first 4 tests and their repetition. On the legend the first number shows the crack number and the second number shows the trial.

The strain field (maximum and minimum normal strains) of the ROI corresponding to the cut length shown in the Figure 77.d can be seen in Figure 80. To allow comparing them to the intact pelvis strain (Figure 74), the color map has been kept constant as can be seen in Figure 79. For simplicity, only the even increments are shown like before.

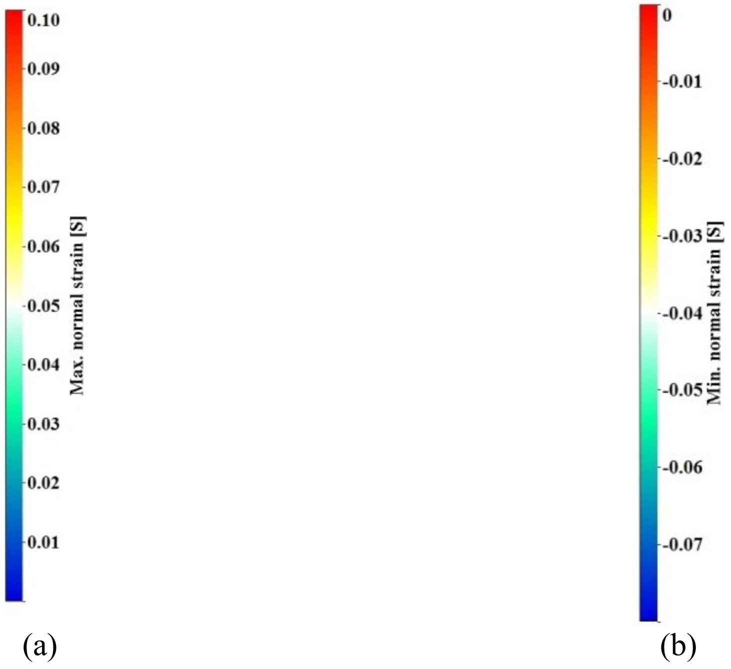
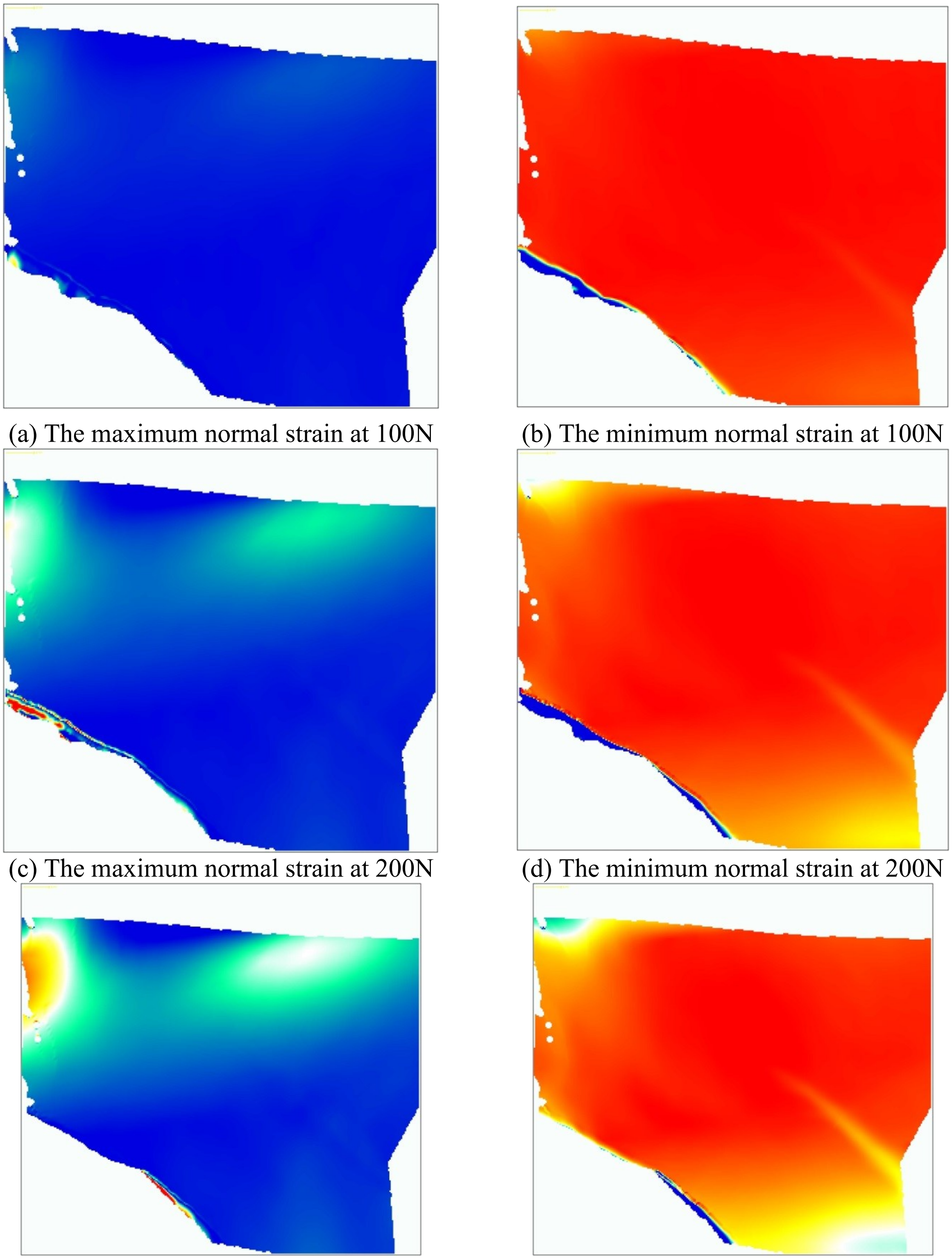


Figure 79: The color map used for strain fields in Figure 80. Figure (a) shows the color map used for maximum normal strain, and figure (b) shows the color map used for minimum normal strain.



(a) The maximum normal strain at 100N

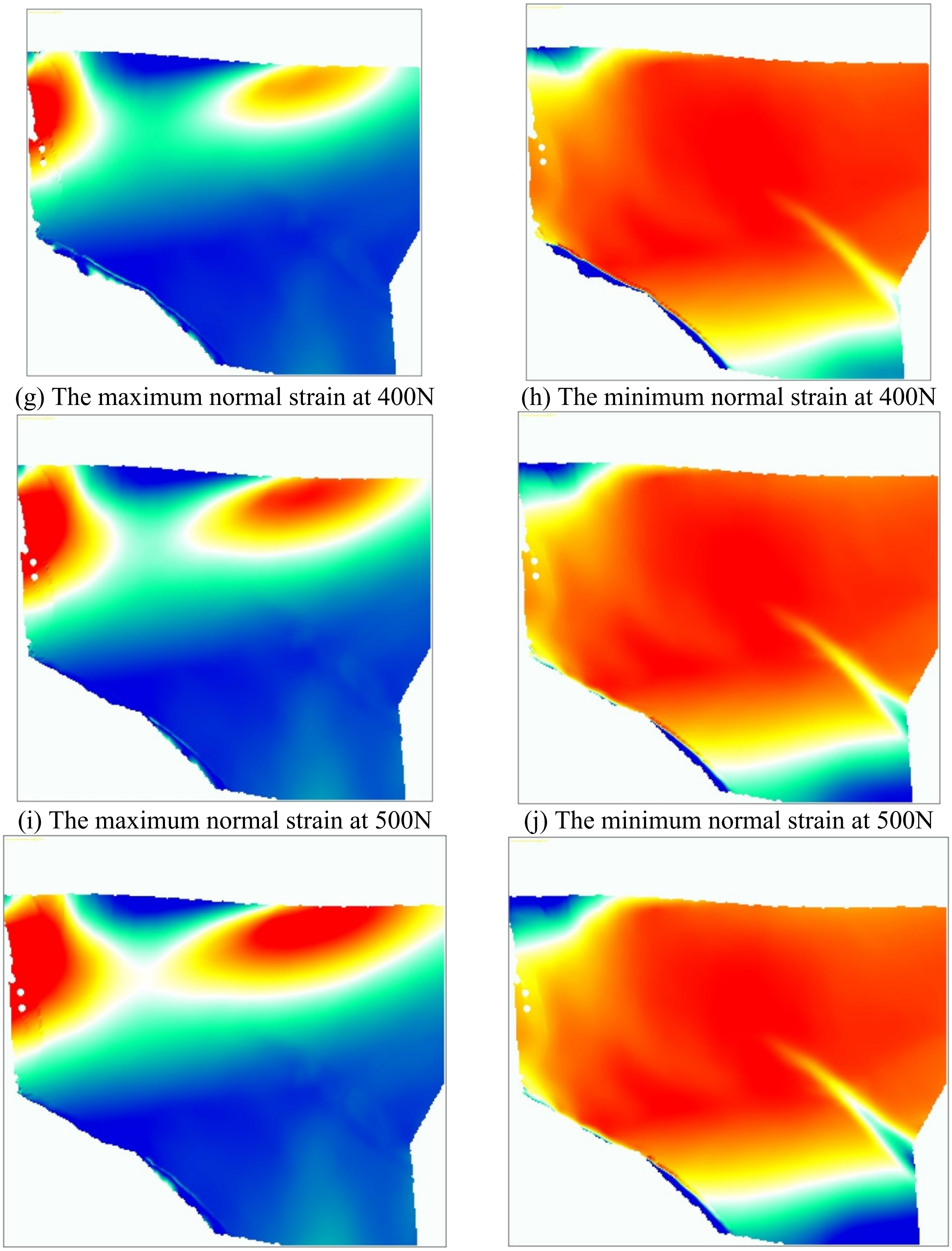
(b) The minimum normal strain at 100N

(c) The maximum normal strain at 200N

(d) The minimum normal strain at 200N

(e) The maximum normal strain at 300N

(f) The minimum normal strain at 300N



(g) The maximum normal strain at 400N

(h) The minimum normal strain at 400N

(i) The maximum normal strain at 500N

(j) The minimum normal strain at 500N

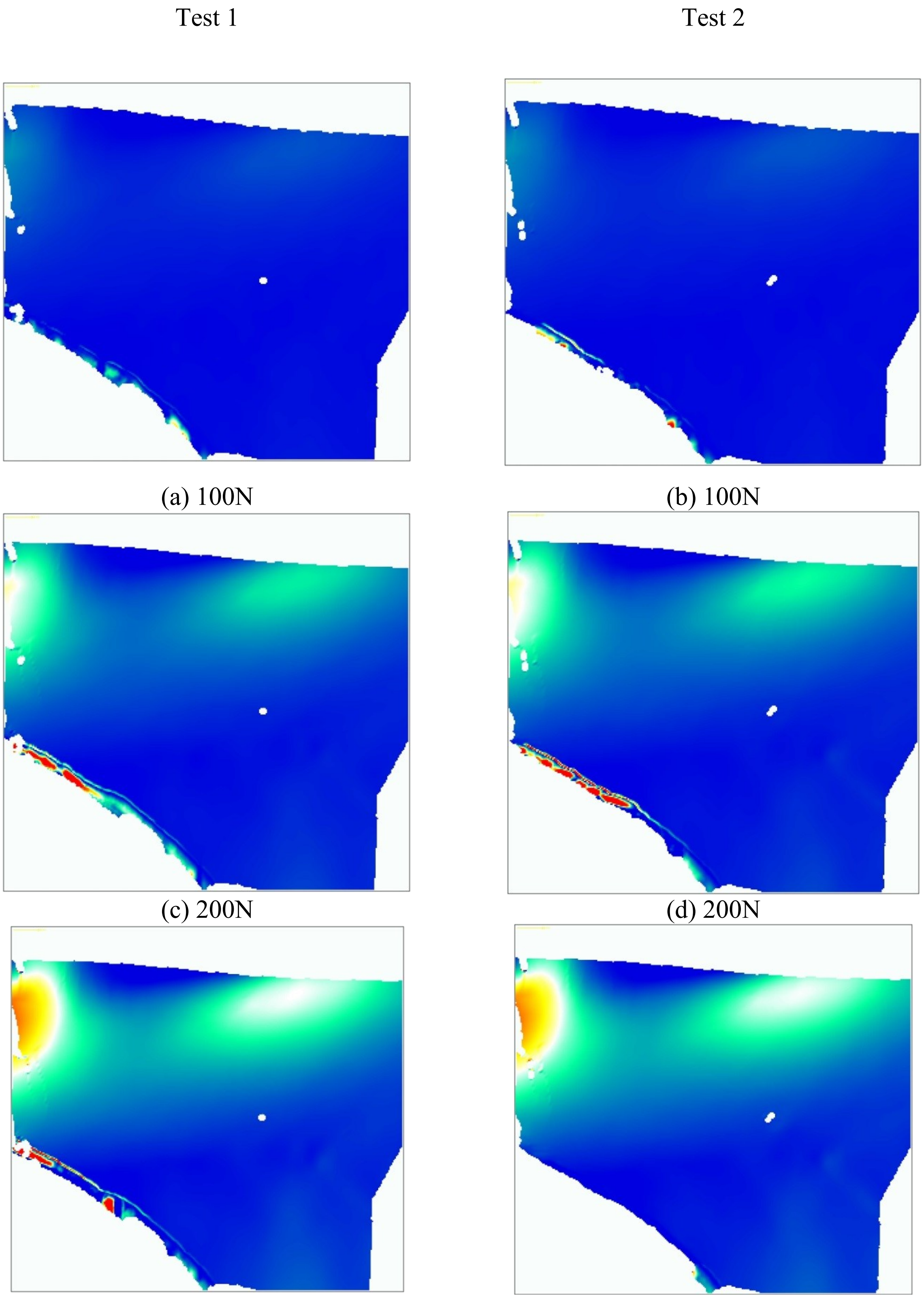
(k) The maximum normal strain at 600N

(l) The minimum normal strain at 600N

Figure 80: The strain field of the pelvis with the cut length seen in Figure 77.d during loading. The left column shows the maximum normal strain for each frame whereas the right column shows the minimum normal strain. Also each row shows a 100 N increase in loading from 100N to 600N.

Comparing Figure 74.k and Figure 80.k, it is clear that the strain is increased. The reason is that because of the cut at the bottom of the region, the strain flow can pass through that area and it needs to pass from the top of the cut which result an increase in the strain.

In order to confirm the good repeatability of the test, two sets of data corresponding to the cut length shown in Figure 77.c are shown and are compared. Figure 81 and Figure 82 show the maximum and minimum normal strain found in both tests. For simplicity, only the even increments are shown like before. It can be seen the strain field of both tests are very similar.



Test 1

Test 2

(a) 100N

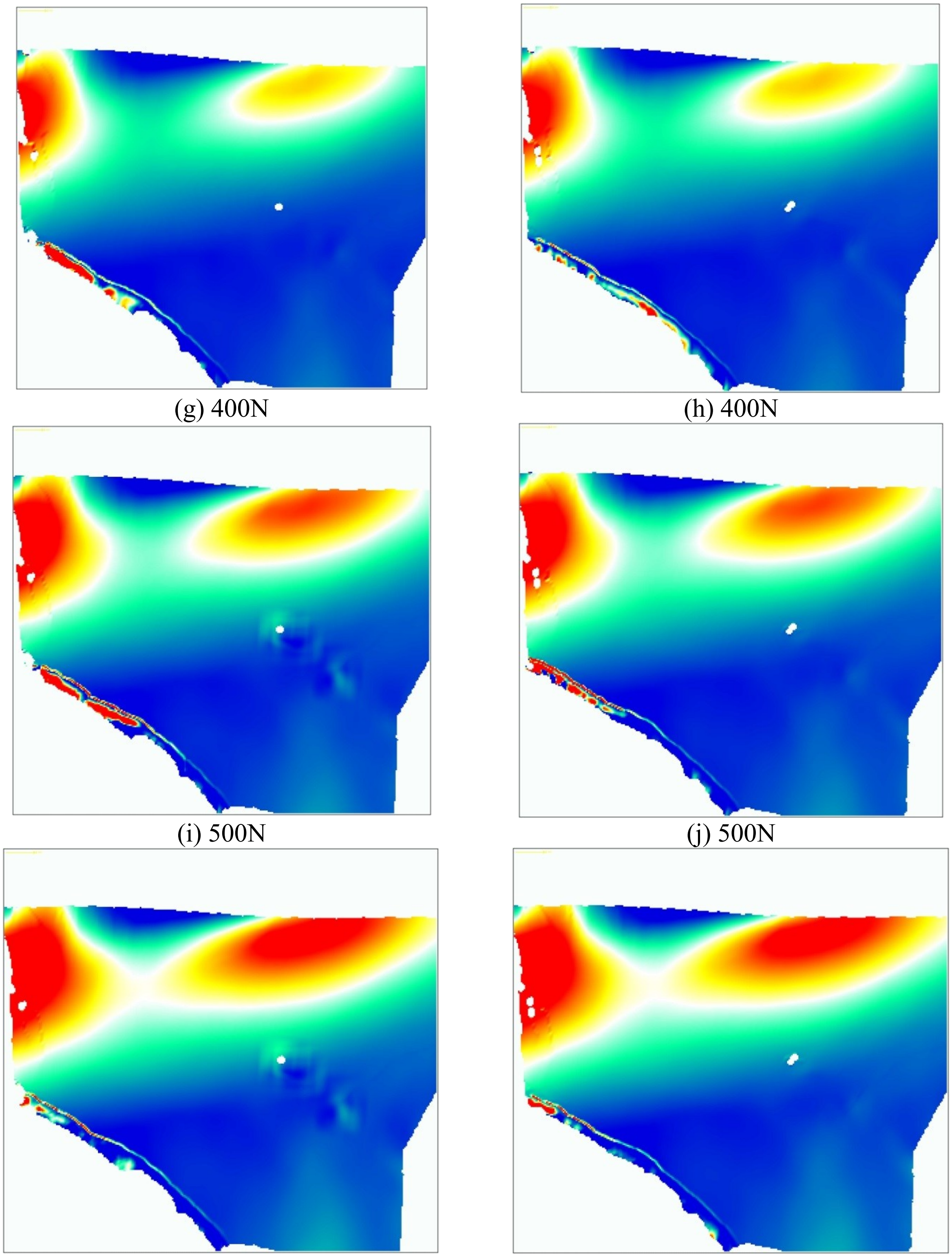
(b) 100N

(c) 200N

(d) 200N

(e) 300N

(f) 300N



(g) 400N

(h) 400N

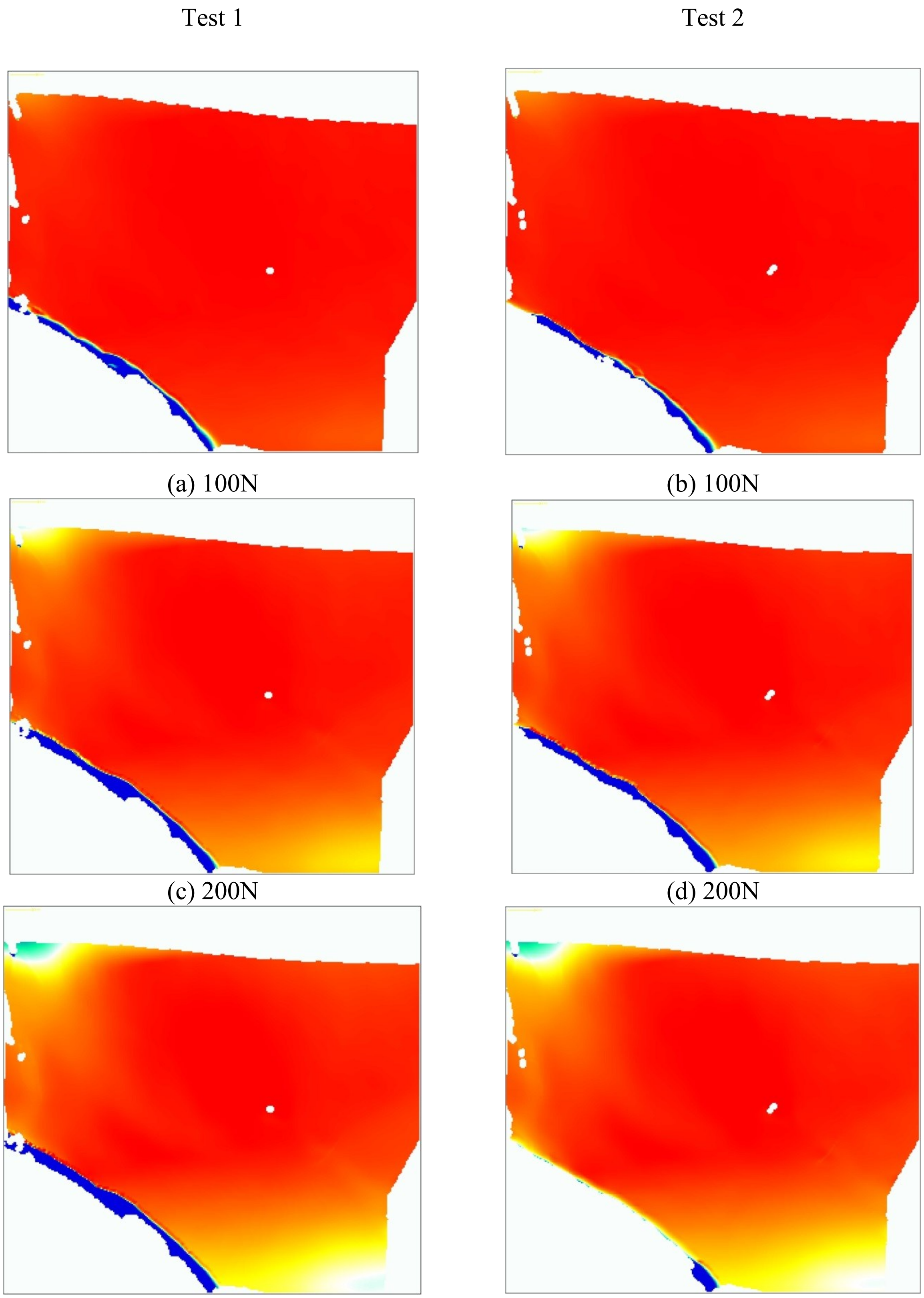
(i) 500N

(j) 500N

(k) 600N

(l) 600N

Figure 81: Maximum normal strain corresponding to the cut size shown in Figure 77.c. The left column show the data of the test 1 and the right column show the data of the test 2. The color map is similar to Figure 79. Also each row shows a 100 N increase in loading from 100N to 600N.



(a) 100N

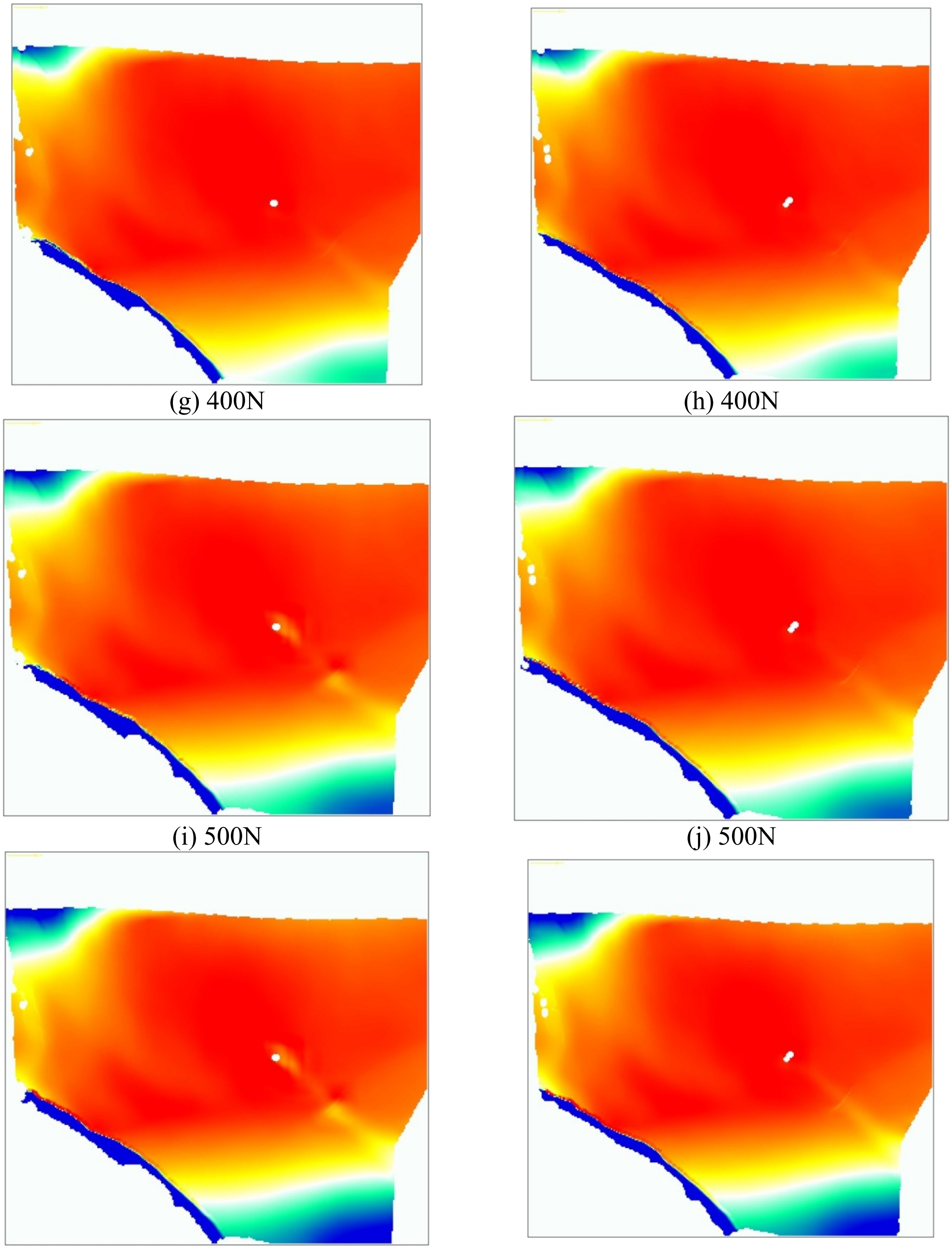
(b) 100N

(c) 200N

(d) 200N

(e) 300N

(f) 300N



(g) 400N

(h) 400N

(i) 500N

(j) 500N

(k) 600N

(l) 600N

Figure 82: Maximum normal strain corresponding to the cut size shown in Figure 77.c. The left column show the data of the test #1 and the right column show the data of the test #2. The color map is similar to Figure 79. Also each row shows a 100 N increase in loading from 100N to 600N.

In order to compare the strain fields more quantitatively, two areas were chosen on the field of view as can be seen in Figure 83. Using the commercial software, the average of strains (both maximum and minimum normal strains) in the areas were obtained and shown in Figure 84 and Figure 85. The figures show that the measured strain on each step (with similar loading) is very similar after repeating the test. These figures plus the Figure 81 and the Figure 82 confirm the very good repeatability of the tests.

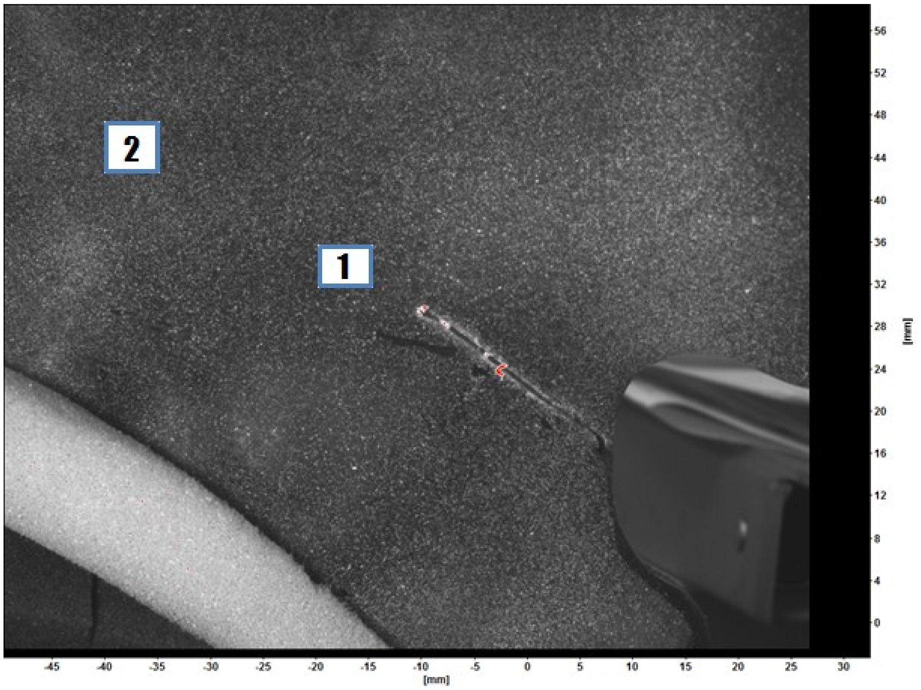


Figure 83: Selected areas for measuring strain

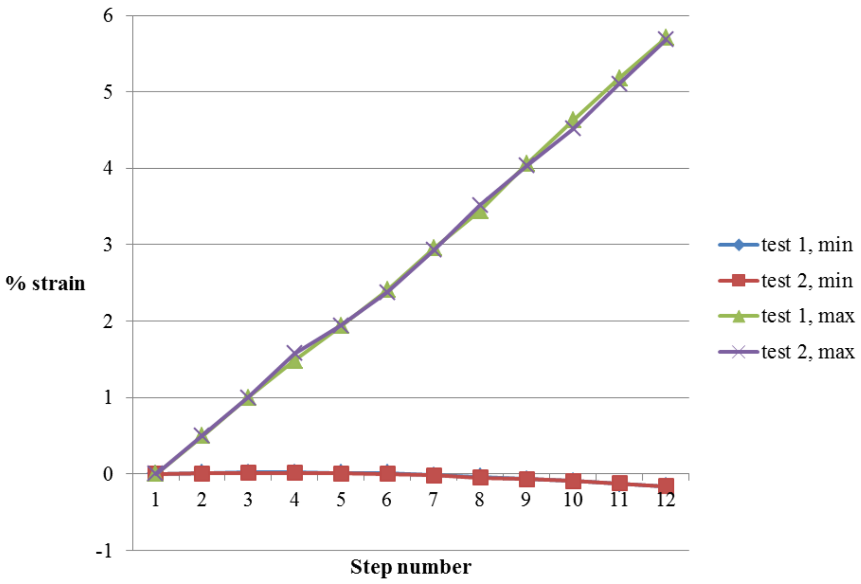


Figure 84: Average strain in the area #1

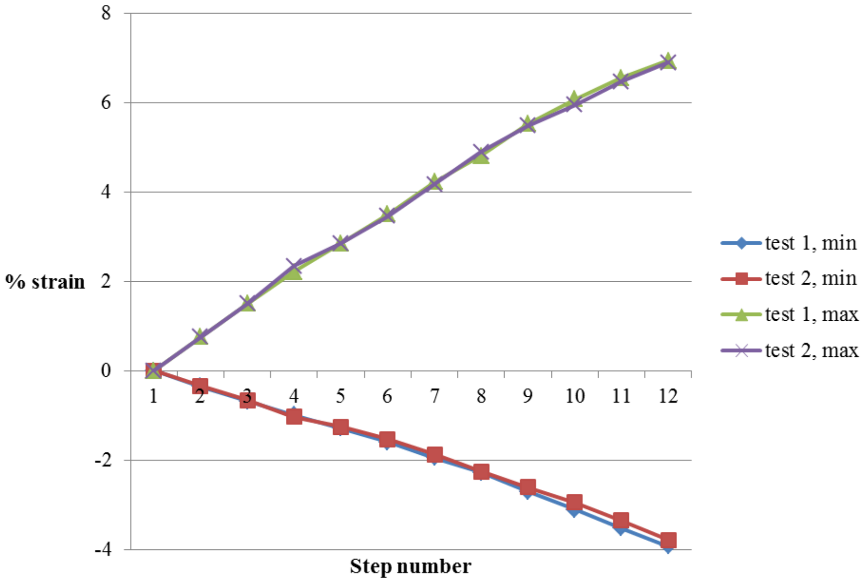


Figure 85: Average strain in the area #2

Table 15 shows the numerical evaluation of the data shown Figure 84 and Figure 85. As can be seen the average error and deviation is small except the minimum normal strain on the area 1. The reason behind the large percentage error is that the values were very small (small values such as: 0.007 or 0.008). Therefore the ratio of the percentage error was obtained large due to dividing by very small numbers. The average difference in the results was 0.005.

Table 15: Evaluation of the repeatability of the test on the intact pelvis. The calculations were done based on the data shown in Figure 84 and Figure 85.

		Average %error	%Deviation
Area 1	**Min normal strain**	27.54	75.51
	Max normal strain	1.65	0.31
Area 2	**Min normal strain**	3.57	0.13
	Max normal strain	1.40	0.25

Up to the 4th cut length (Figure 77.d) no gap opening was observed visually with the highest load of 600N as can be seen in Figure 86. The reason behind this observation was initially assumed to be the small length of the cut in the region. Since the strain of the healthy pelvis was low in that region, not much loading was being transmitted through that area. So the crack was not being opened, and it was concluded that the gap opening was too small to be seen with the naked eye. Accurate measurements confirmed later that there was a 12 µm opening in the gap.

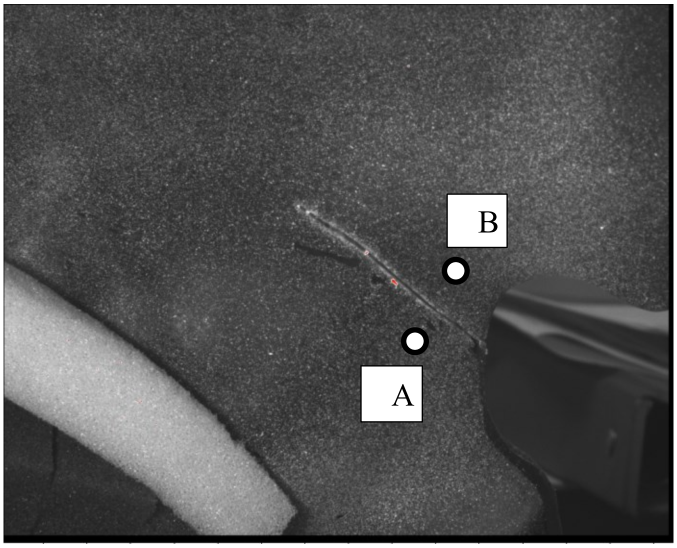


Figure 86: The raw image of the 4th cut with the loading of ~600N

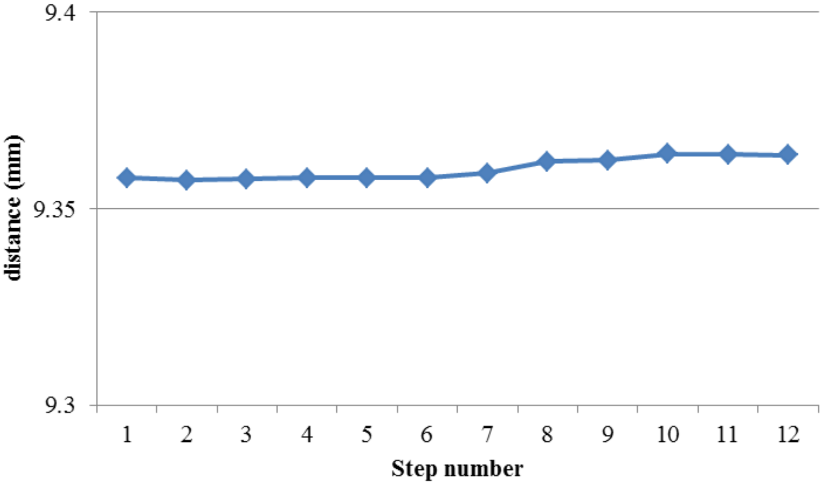


Figure 87: Gap opening (distance between the points A and B shown in Figure 86)

During the test on the 5th cut length, an observation was made. As described before, the only loading component was the vertical force; however, the load cell showed a horizontal force and other moments as well as can be seen in Figure 88.a. This important observation made clear why there was not gap opening. One side of the hemi-pelvis was fixed in the pot, and the other side on which the femur was pushing had just one degree of freedom. Therefore, even though the gap opening was the natural response of the hemi-pelvis, there was no room for this displacement. This moment on the coronal plane was recorded before altering the loading and can be seen in Figure 89.

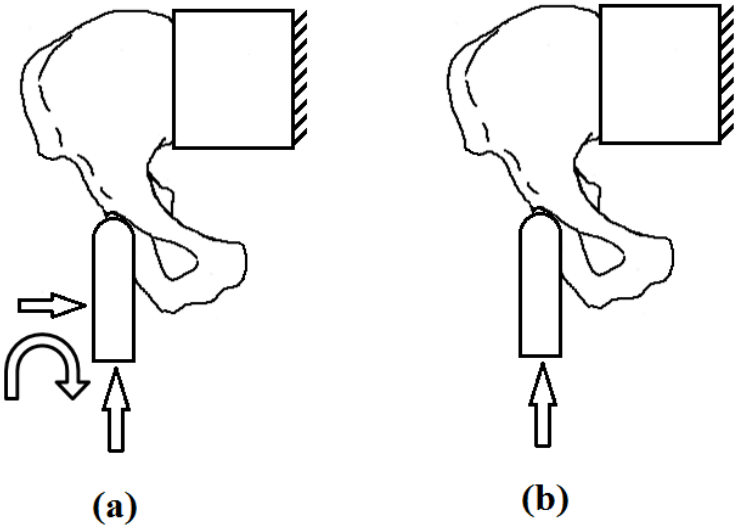


Figure 88: The BC used in the tests. Figure (a) shows the initial BC up to the 4th cut length, and figure (b) shows the desired, BC which was achieved after the 4th cut length

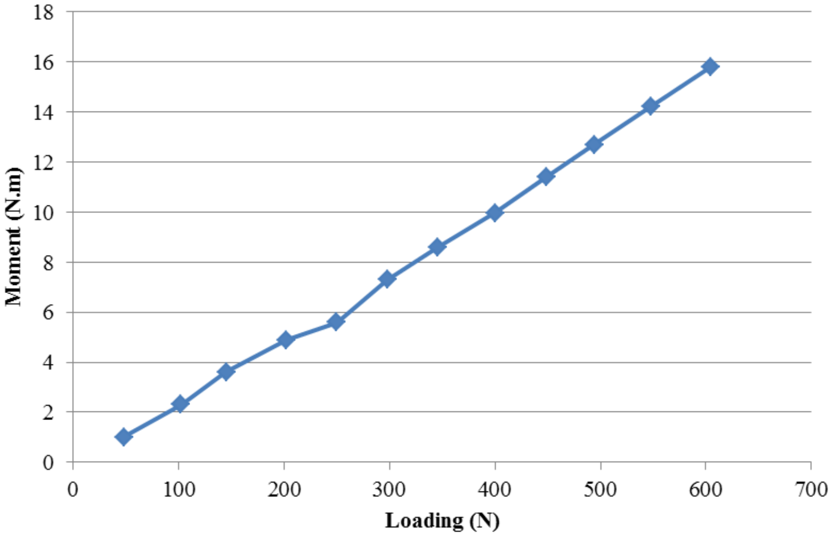


Figure 89: Increasing moment on the coronal plane while increasing the loading corresponding to the 5th cut length shown in Figure 77.e

Thus, on the next test, the femur was displaced horizontally to reach a state with only vertical force (Figure 88.b). With this new testing condition, which was intended to happen to begin with, the gap opening was visually seen. Nonetheless, during this test, due to a sudden overloading at the 6th step, a crack was generated in the test piece. Figure 90 shows the unintentional crack emerged in the hemi-pelvis. The crack was exactly generated toward the area with considerable amount of strain which was shown in Figure 81.k.

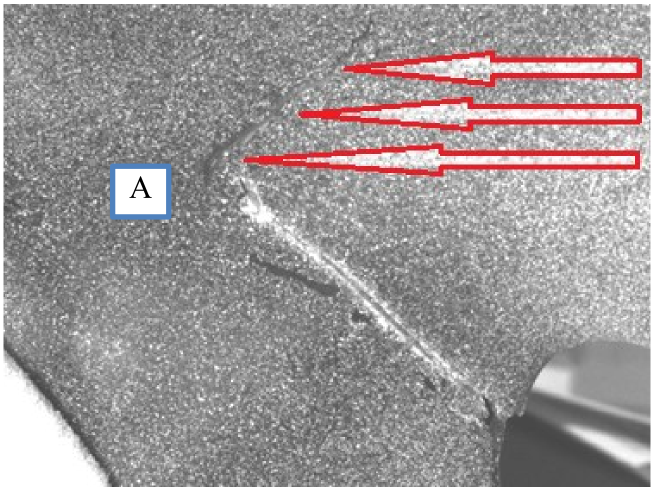


Figure 90: Unintentional crack generated during the 5th set

Since the test with the right loadings and the gap opening was repeated and the data related to the 4 steps without the gap opening was available, there was a chance for comparing the effect of the loadings which can be seen in Figure 92 and Figure 93. They show a dramatic change in the strain field demonstrating the importance of the BC on testing with the identical loadings.

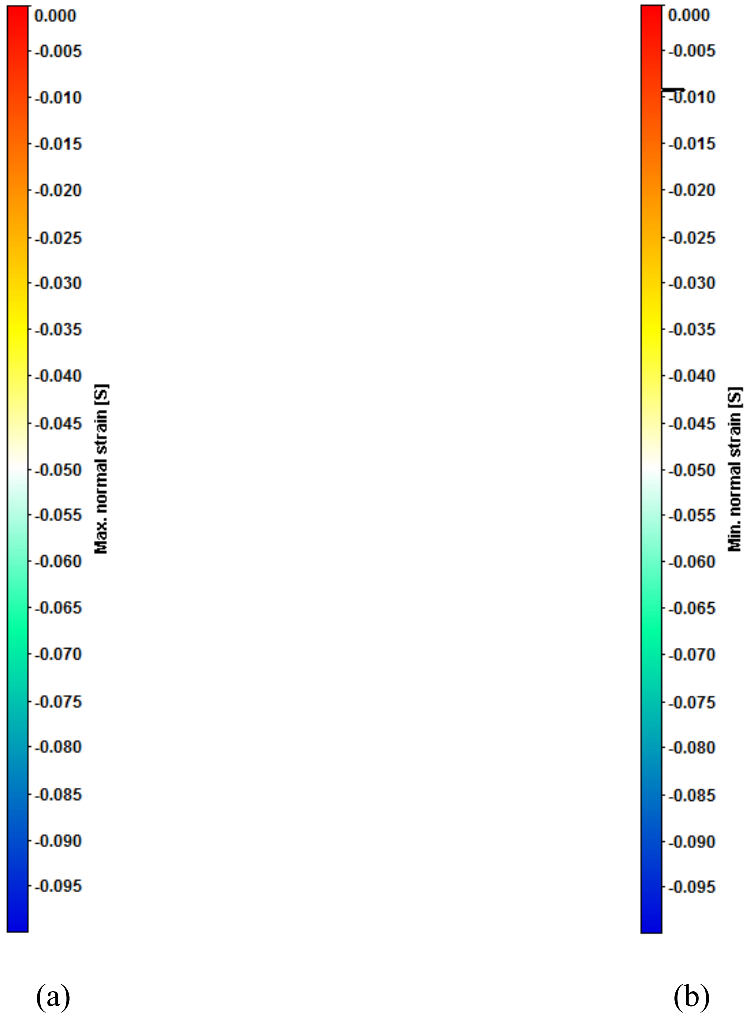
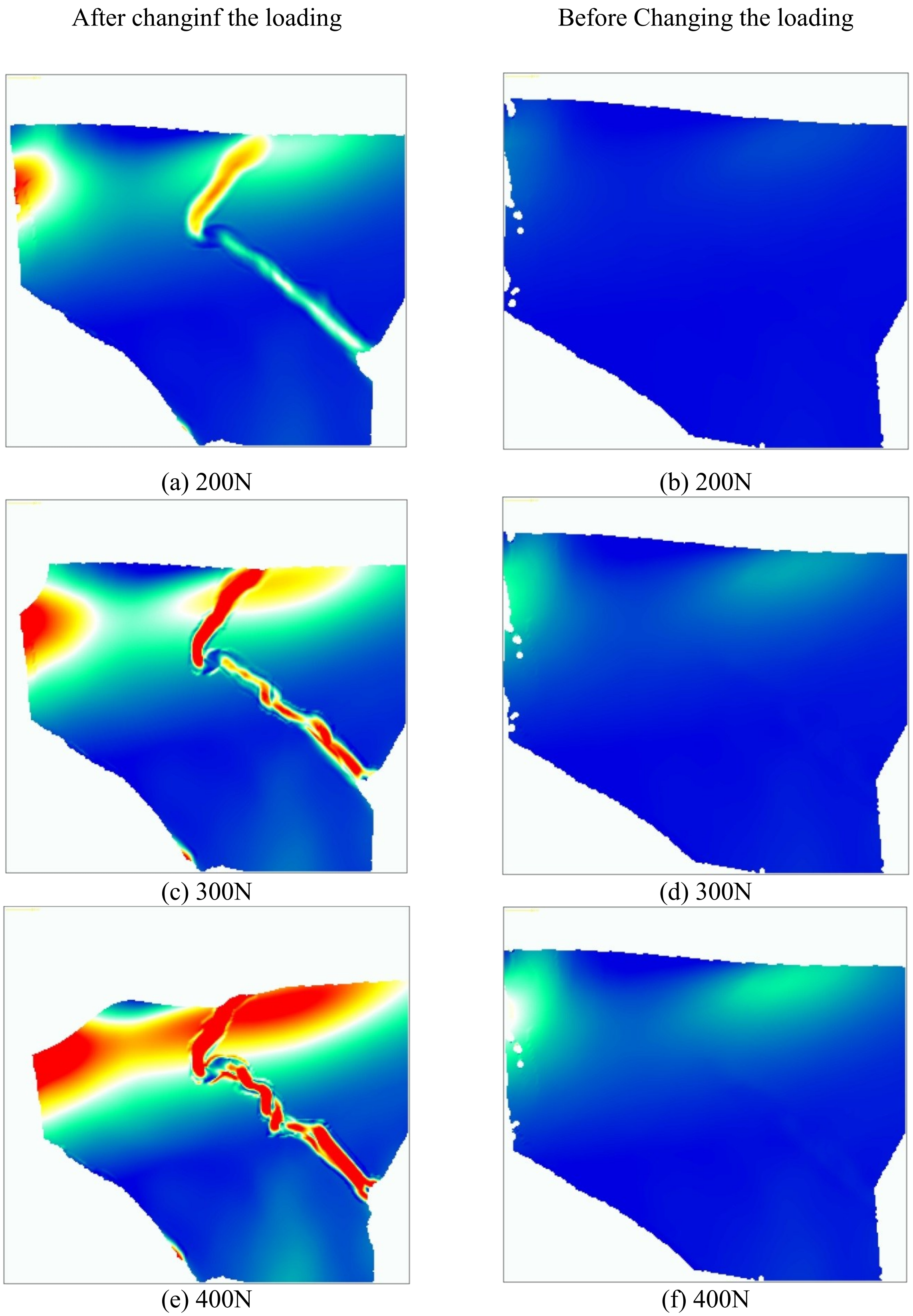


Figure 91: Color map used in Figure 92 and Figure 93. The color map (a) is used for maximum normal strain and the color map (b) is used for minimum normal strain



After changinf the loading

Before Changing the loading

(a) 200N

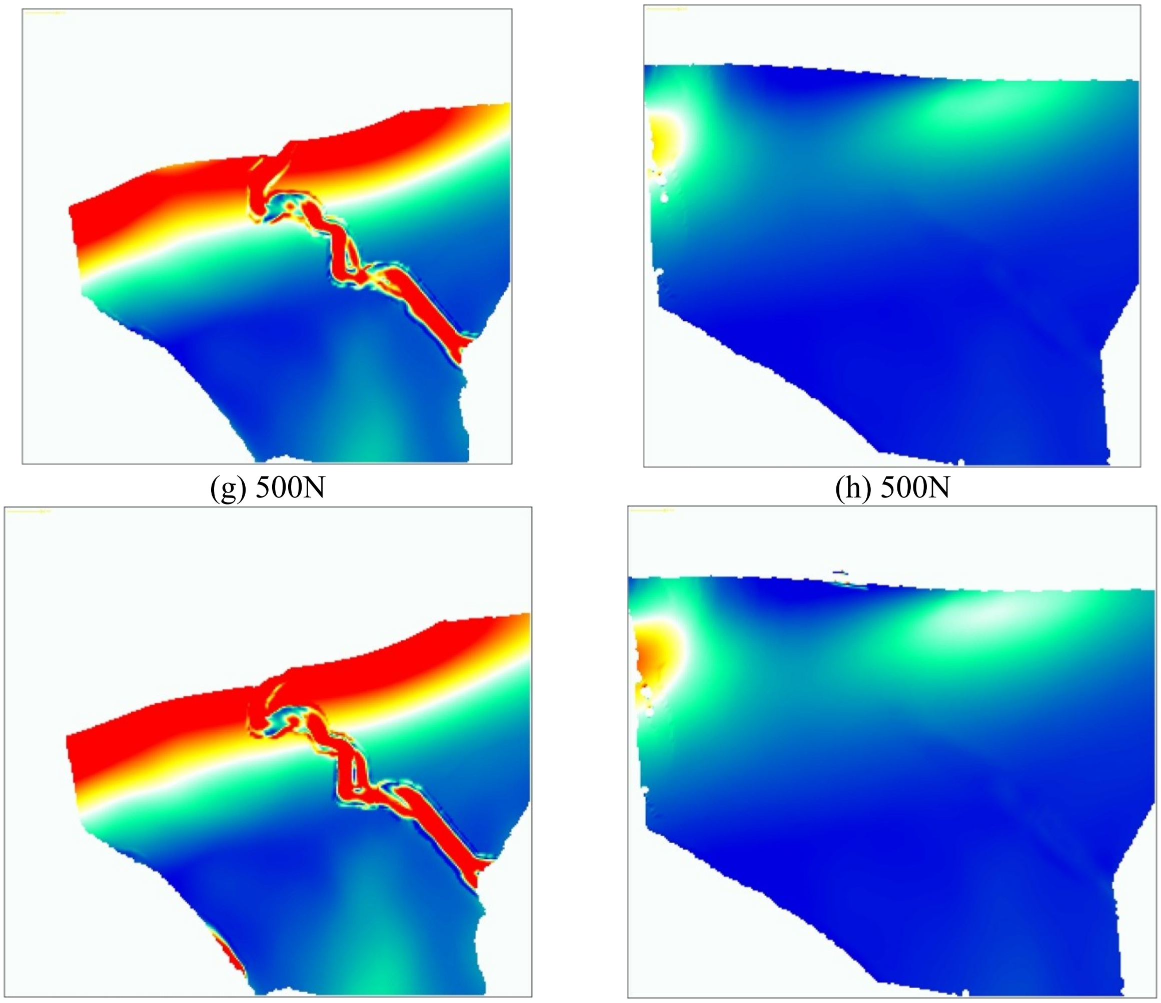
(b) 200N

(c) 300N

(d) 300N

(e) 400N

(f) 400N



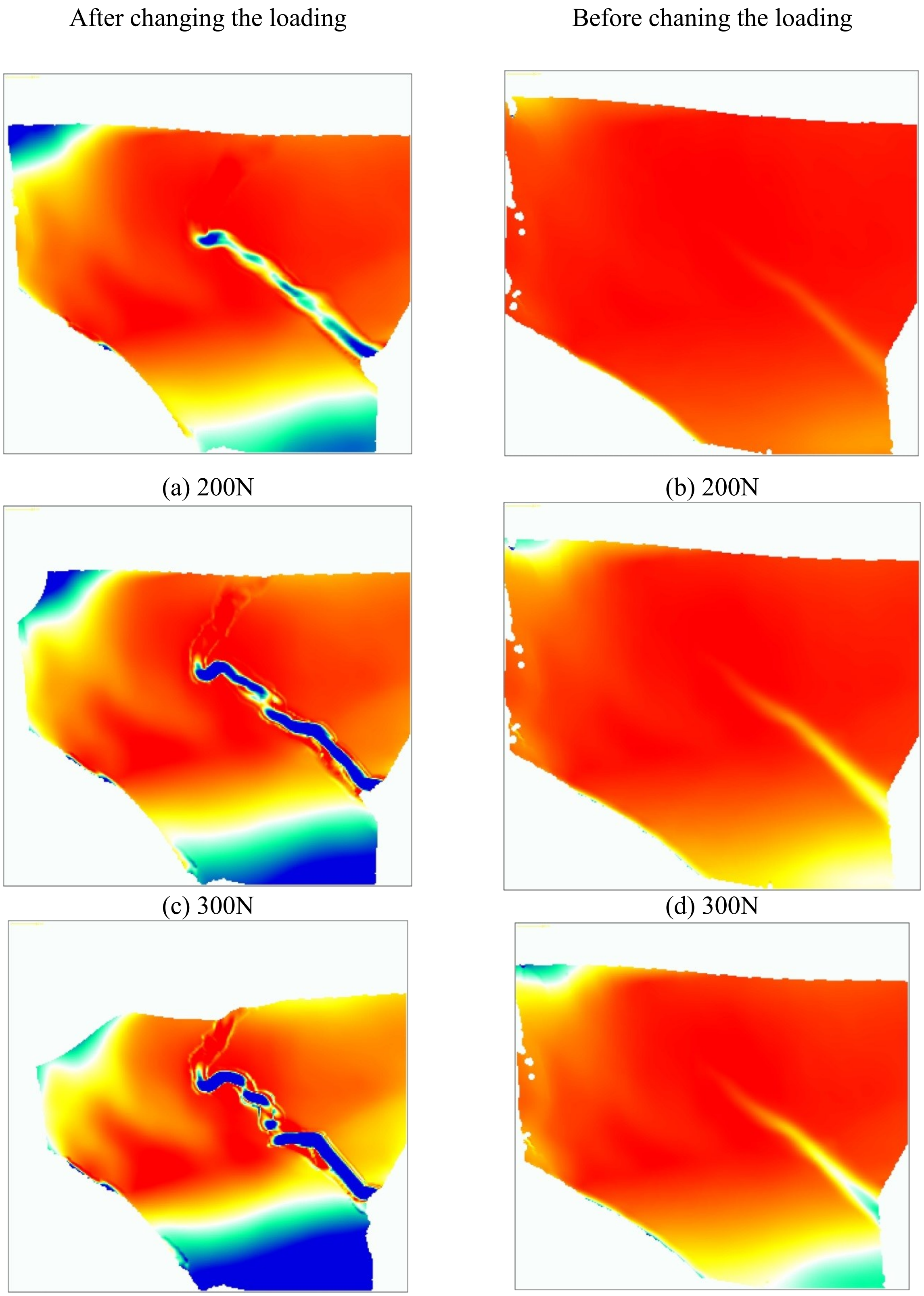
(g) 500N

(h) 500N

(i) 600N

(j) 600N

Figure 92: Maximum normal strain before (right column) and after (left column) changing the loadings. Also each row shows a 100 N increase in loading from 200N to 600N.



After changing the loading

Before chaning the loading

(a) 200N

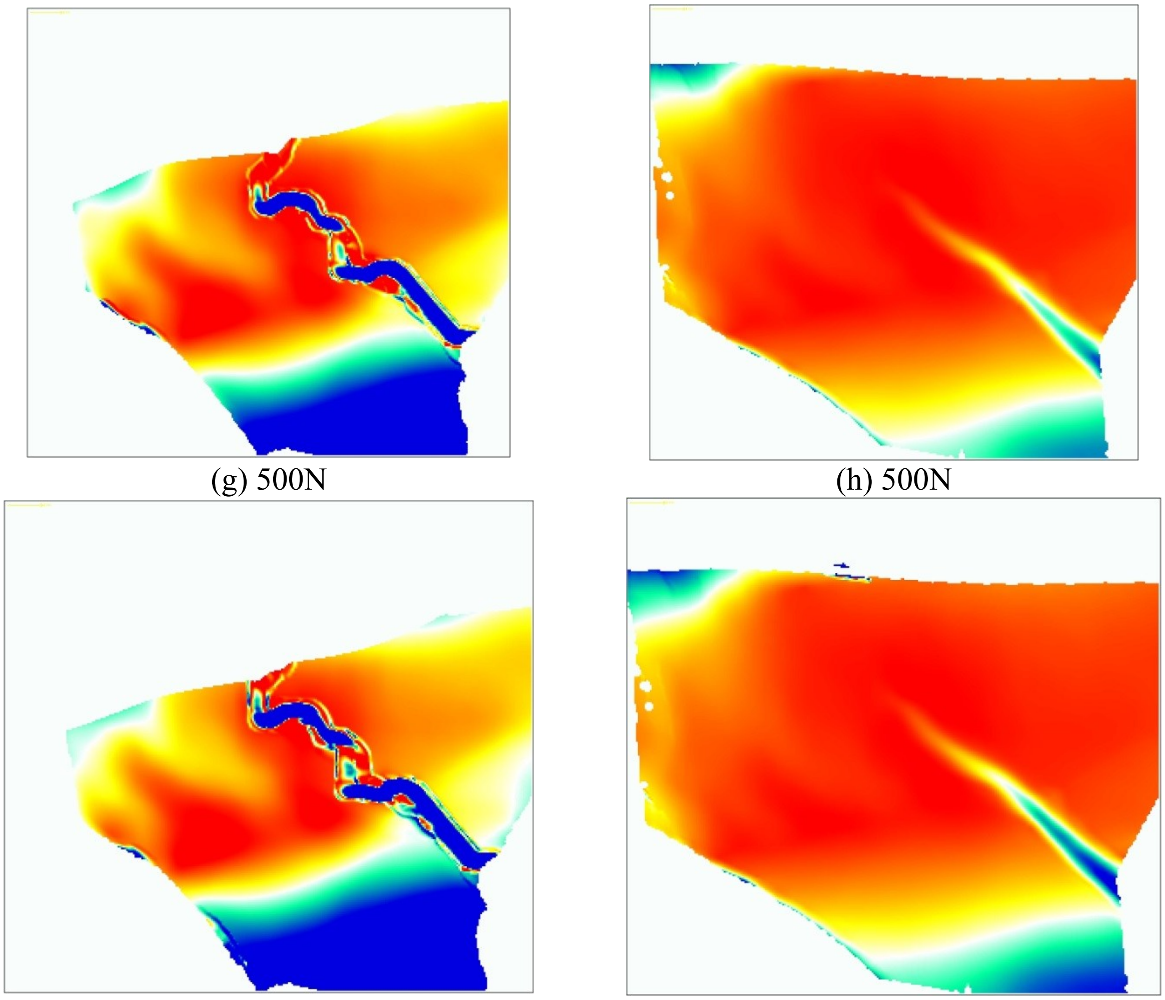
(b) 200N

(c) 300N

(d) 300N

(e) 400N

(f) 400N



(g) 500N

(h) 500N

(i) 600N

(j) 600N

Figure 93: Minimum normal strain before (right column) and after (left column) changing the loadings. Also each row shows a 100 N increase in loading from 200N to 600N.

For a quantitative evaluation of the strain during the loading, the area "A" was selected on the hemi-pelvis as can be seen in Figure 90. The maximum and minimum normal strain in the region is shown in Figure 94.

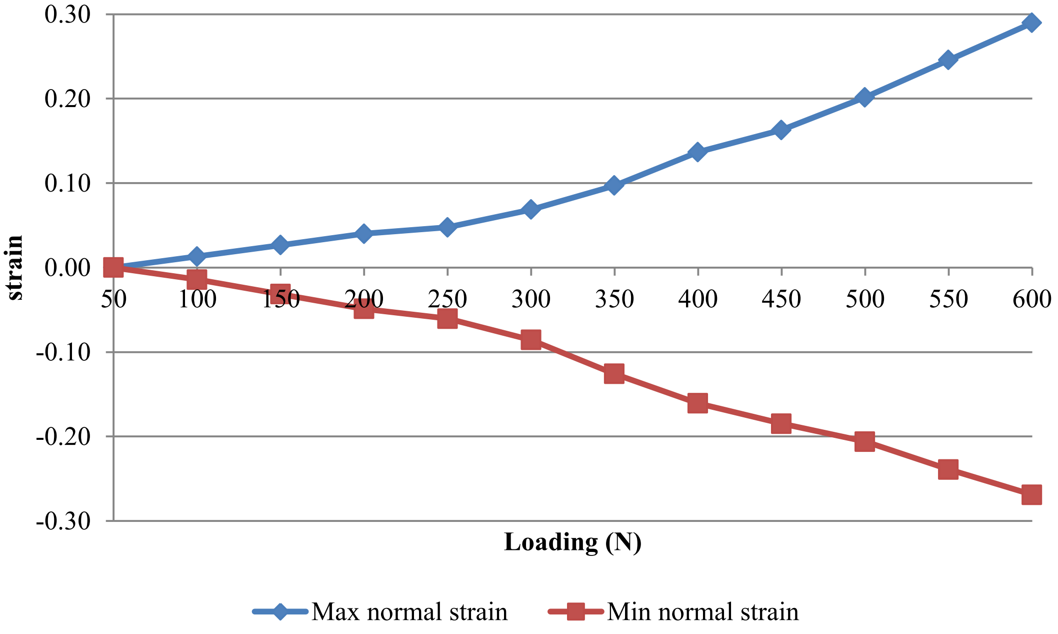


Figure 94: Maximum and minimum normal strain of the area A shown in Figure 90

The test was carried on with increasing the cut length. It was continued until the cut reached the edge of the field of view. Gap opening as well as its mode of opening was measured using the Matlab programs at the end. To make sure that the measurement is not affected by a point with singular error, the measurements were repeated for three adjacent points. For measurements two points were chosen to find the change in distance between them which are mentioned in the figure as "couple". Figure 95 shows the selected points (couples) and the defined coordinate system. The in two areas "1" and "2" the measurements were taken on a different couple of points as shown in Figure 95.

The same behavior in all couples in the groups 1 & 2 shows that those were not points with singular error. Therefore, it is possible to investigate the motion in each. Figure 99 shows the displacement of the blue couple.

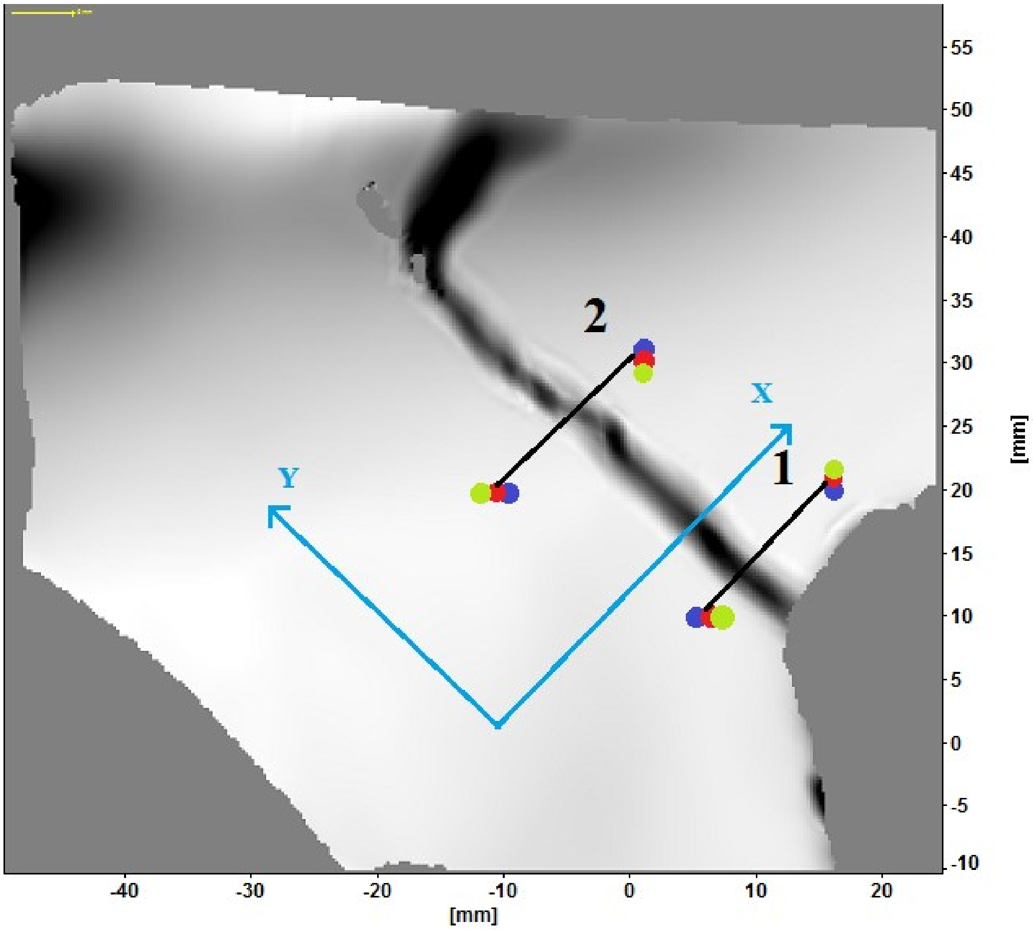


Figure 95: The selected points and coordinate system for measuring the main gap

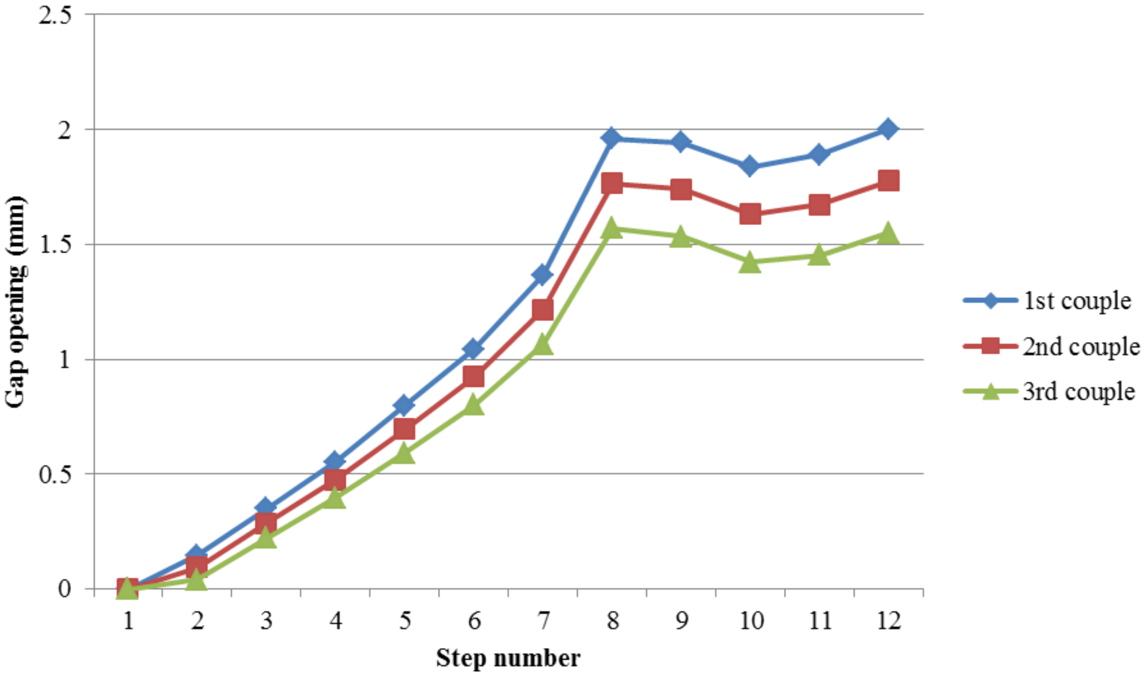


Figure 96: Changes in the distance of the points in group "1". The 1st step was at 50 N with increasing of 50 N in each up to 600N in the 12th step.

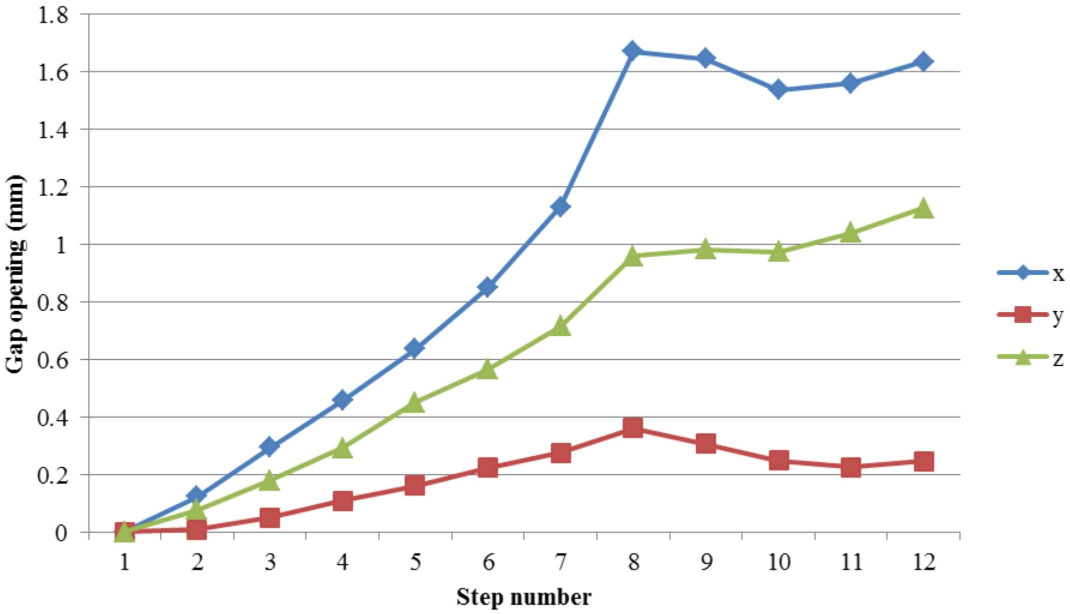


Figure 97: The components of the opening in the blue couple of group "1". The 1st step was at 50 N with increasing of 50 N in each up to 600N in the 12th step.

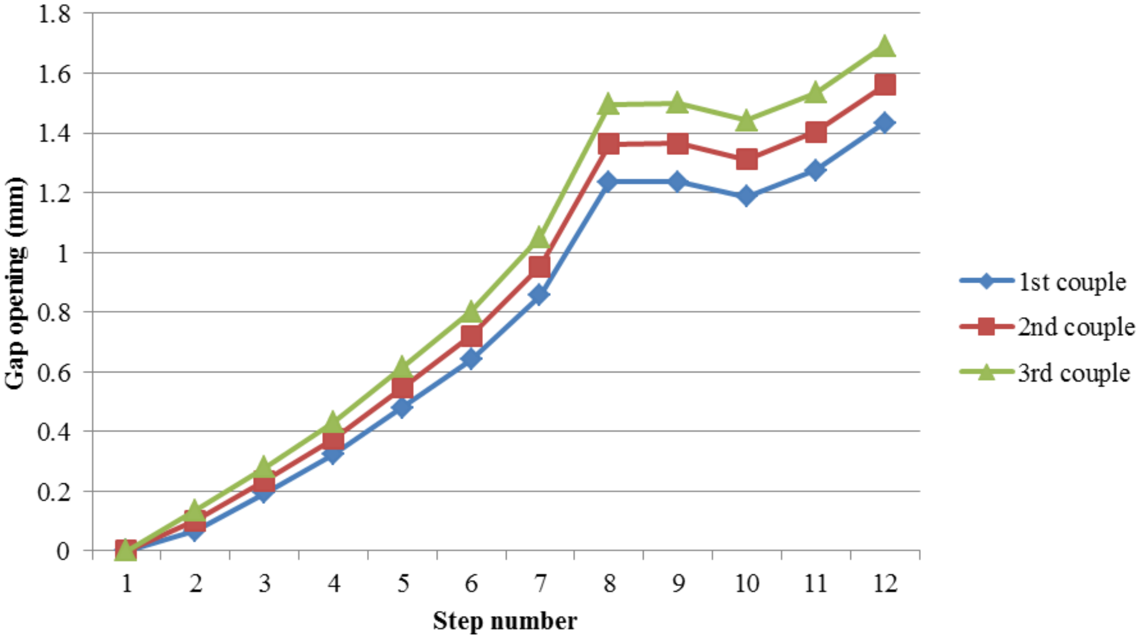


Figure 98: Changes in the distance of the points in group “2”. The 1st step was at 50 N with increasing of 50 N in each up to 600N in the 12th step.

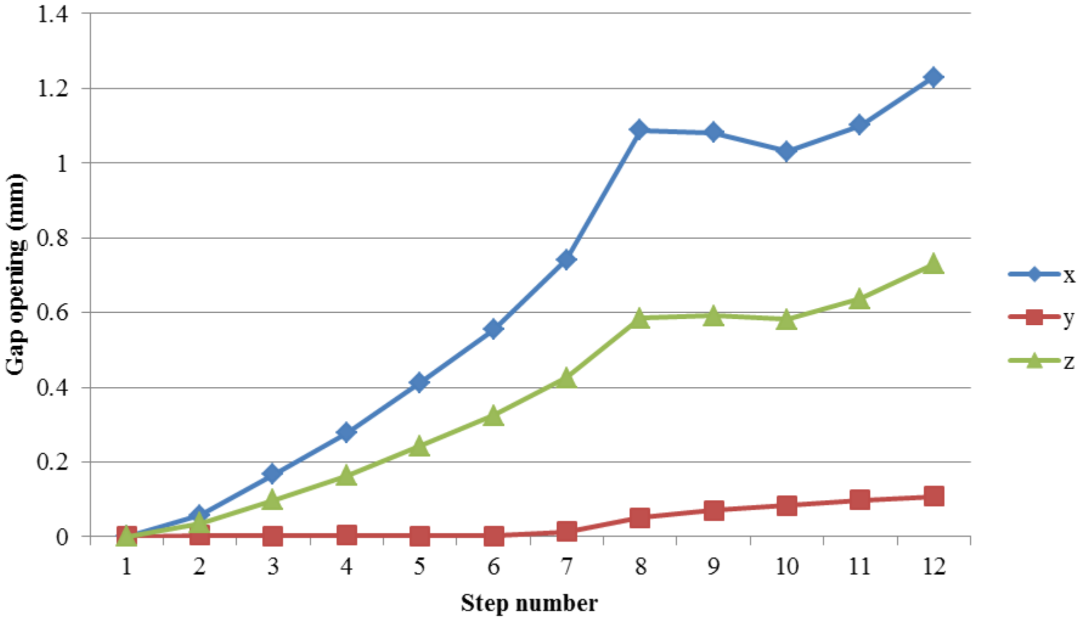


Figure 99: The components of the opening in the blue couple of group “2”. The 1st step was at 50 N with increasing of 50 N in each up to 600N in the 12th step.

This procedure was again repeated for the short unintentional crack. Three couples of points were chosen on both sides of the crack, and a new coordinate system was defined as can be seen in Figure 100. The coordinate system was defined in a way that opening in the *X* represents the mode I. Due to the fact that this region was too close to the border of the field of view, after the 9th loading step the points could not be detected by the software and the data for this region was available up the 8th loading step (400N). Again the distances between three couples were compared to make sure that there was no singular error. Figure 101 shows the gap opening measured using each couple of points.

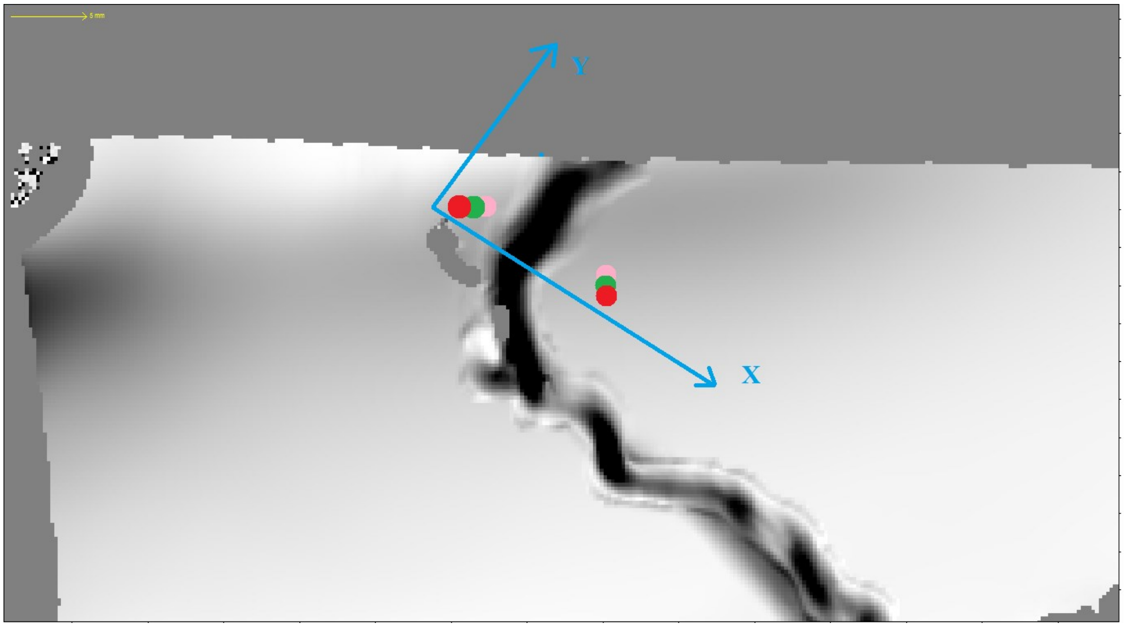


Figure 100: The selected points and coordinate system for measuring the short crack

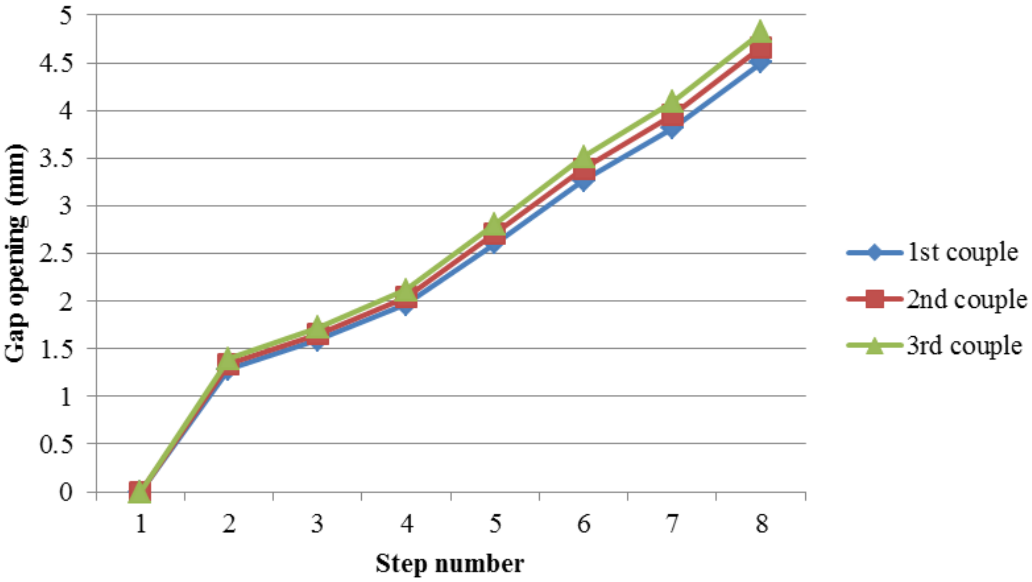


Figure 101: Changes in the distance of the points shown in Figure 100

As can be seen in Figure 101, measuring the gap opening by each couple led to very similar result assuring that there is no singular error in any of them. Therefore, it is possible to study the gap opening mode by each. Using the blue couple, Figure 102 was obtained.

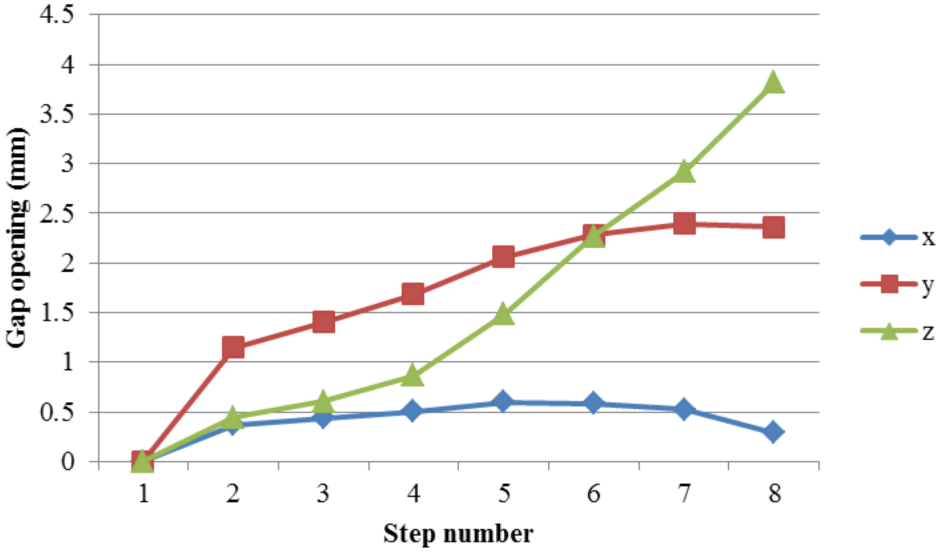


Figure 102: The components of the opening for the blue couple

In this chapter the results were shown and illustrated the behavior of a broken hemi-pelvis. The setup has been able to identify the strain field on the pelvis. Even though there are no other data to compare the pelvis results, the results can be trusted as the apparatus has been tested and verified in multiple stages described in chapter 3. The strain field on the surface varied during the loading as well as when the cut length rose. Also the gap measurement in 3D showed the compound behavior of mode I and III for the long cut and mode II and III for the short crack.

Chapter 5: Discussion

I. Strain Field

The strain fields for both intact and broken hemi-pelvises were obtained. Figure 74 shows two regions of higher strain one at the top of the acetabular and the other one close to the sacrum. The positions of the regions are similar to the data found in the literature [38], [39]. However, the position of the area “A” is a bit different. The reason is that in both cases (the current and mentioned studies) the area above the femur is showing higher strain, and since the direction of loading is different, their location is different too. The other difference is that the region closer to the sacrum was found as with the higher strains in previous studies, but in Figure 74 the area at the top of the acetabulum shows higher strain. The reason behind this fact could be both the different direction of loading and the different support at the pubis. They also found the strain value of 1.0ε in the regions. In addition to the different support for the pubis, with the different loadings (600 N in the current study 1600N in the mentioned), and with the different direction of loadings, the results should be different, but at the same range.

The repeatability of the test was assessed using the average percentage error and its deviation as shown is Table 15. The maximum normal strain shows an excellent repeatability with the average error of 1.65% and 1.40%. The average error percentage of the minimum normal strain on the area 2 was 3.57%, which is acceptable, but is 27.54% in the area 1. The reason behind this observation is the small value of minimum normal strain in the area 1. Such small values (0.007 and 0.008) are close to the minimum accuracy of the system. So it is not a good measure for evaluating the repeatability. Nonetheless, the average difference was 0.005 which shows that the difference remained small during the test.

Increasing the strain in the broken pelvis seen in Figure 92 shows that the pelvis is considerably weakened by the crack. Also, the larger strain values (Figure 94) in the broken pelvis show that it is prone to external loading with the possibility of crack growth in other directions toward the areas of higher strain. This issue was observed with the unintentional crack in Figure 90. The crack was generated growing toward the area of high strain at the sacrum. Figure 92.e demonstrates how the crack path lies on the high strain region.

The crack, however, did not change the position of the regions with the higher strain. As can be seen in Figure 76.e two areas above the femur and close to the sacrum remained with highest strain values. The same can be said for the minimum normal strain in Figure 93. The only difference visible in the strain fields of Figure 92 is an area with higher strain in the lower middle of Figure 92.e - Figure 92.i around 20 mm next to the acetabulum cup. This area did not show a different strain value from the nearby areas. The explanation of this change is that the materials on the crack sides do not undergo a tension anymore. So any strain flow because of the tension found in the lower areas, has to pass from a certain distance from the crack which created the higher strain in the middle lower area.

The other important observation is that the minimum normal strain had a segment increase compared to the maximum normal strain. Figure 75 shows that the maximum and minimum normal strains are around 0.11 and -0.03 resulting in 0.14 as the Mohr's circle diameter, but these number changes to 0.3 and -0.in respectively resulting in 0.58 as the Mohr's circle diameter. With a less than threefold increase in the maximum normal strain, the Mohr's circle diameter increased by 6 six fold. So the huge increase in the minimum normal strain had a huge effect on the strain state. Investigating the strain field of the broken pelvis is a novel research carried out in this work.

II. Gap Opening

The gap opening was measured using the DIC system. This has not been done in other studies on the pelvis and is considered a novel aspect of this study. Using DIC allows investigating of the opening mode.

Figure 96 and Figure 98 show that the gap is increased until the 8th step with the loading of ~400 N. Thereafter, the distance between the points on the surface was not increased considerably and even was decreased in some steps. This observation suggests that the crack is too complex to behave linearly especially because of the existence of the second gap (unintentional one) and the complex geometry of the hemi-pelvis. As expected, the maximum opening in the group 1 was higher than in the group 2 simply because of more distance from the tip around which the part rotates partially.

Figure 97 and Figure 99 show that the gap was increased mostly in the *X* direction which resembles mode I. Displacement was also seen in the *Z* direction resembling the mode III. Again, this observation confirms the compound opening of the gap. Figure 97 and Figure 99 show the dominant mode I toward which the bone shows the maximum resistance [74].

Figure 102 shows that at the 6th step (300 N) the gap opening in *Y* and *Z* directions are equal. Thereafter, mode III is the dominant with a continuous sharp rising. Opening in the *X* direction is very small in comparison to the other directions and falling after 6th step. Generally, the dominant mode of opening is a compound of II and III. So if a fixation is to be designed to stabilize the broken pelvis, it should minimize the opening on the small crack as it opens in the most destructive modes.

The potential sources of error could be illuminating, speckle size, chosen subset size for image processing, accuracy of images taken by the cameras, possible moving of the camera setup between the calibrations and doing the test. As shown in chapter 3, the error of measuring the strain with a similar condition was 6%.

Chapter 6: Summary and Conclusion

I. Summary of Findings

The goal of this study was to investigate the strain field of a broken hemi-pelvis during loading. Also, measuring the 3D displacement of the fragments was imperative. To test a prosthetic pelvis bone (sawbones), a multi-axis biomechanical testing apparatus (MABTA) was modified. MABTA was equipped with a DIC system, and it was evaluated. The strain field on an intact and a broken hemi-pelvis was obtained, and the 3D displacement of the broken part was also studied. Having the aforementioned results, the objectives discussed in chapter 1 are achieved.

The controlling program of the MABTA was modified. The errors were evaluated where the load cell had the accuracy of 2.5 N up to 2000N loading with 1% error. Also the loading orientation exerted by the femur prosthetic was studied and the necessary jig, to hold the hemi-pelvis on the desired orientation, was designed. The set-up was coupled with a 3D DIC system and was evaluated. The camera set-up detected movements with the error of less than 3.5%.

Moreover, both the repeatability and the accuracy of the strain measurement system were assessed. With a test on a rubber sample, the system showed promising repeatability of the obtained strain field shown on the entire field and compared quantitatively with two points. Then the accuracy of strain measurement was assessed with the average error of 6%.

Finally, tests on an intact and broken hemi-pelvis were carried out. The results of the intact pelvis were similar to the previous studies, and the results of the broken hemi-pelvis showed that the location of the regions with higher strains was the same as the intact hemi-pelvis, but a significant increase in the strain was observed, especially in the minimum normal strain. The other notable change in the strain field was observed at ~20mm to the side of the acetabulum.

Also, in order to study the inter-fragmentary movements, two Matlab programs were developed and tested. The accuracy of measuring the in-plane gap opening was 5μm and the accuracy of measuring the out-of-plane opening was 30 μm. Using the programs on the broken hemi-pelvis showed that on the main cut the dominant mode of opening was found to be mode I, whereas the dominant mode was mode II and III.

II. Possible Improvements and Future Work

Installing the camera system for DIC required proper support. The plates used for this purpose had limited positions. One improvement for using the set-up is developing a moving support to cover any desired angle to look at the hemi-pelvis since the hemi-pelvis needs to be potted on a specific position and orientation.

Improvements to the MABTA control system to make it force controlled and not just displacement controlled. Also, having an emergency stop, when reaching a critical loading, could be added. The other feature could be a history output to save all the data during a test on a separate file as an archive.

The mold in which the hemi-pelvis was potted had barely room to fit and hold the hemi-pelvis in. A bigger mold for this purpose would make the potting easier given that it requires designing the jigs for potting with the same orientation but fitting the mold.

There are multiple potential projects and future work that could be done using FEA to simulate the tests described in this thesis. Utilizing an available CAD model of the hemi-pelvis with the cortical and cancellous bones from the Sawbones Company, and an FEA commercial software would be a valuable. Basic modeling could be done in ABAQUS without much effort. However, modeling a broken pelvis could be challenging, especially with the general weakness of the FEA software to simulate the cracks.

Using the current apparatus, it is possible to work on developing novel fixations. As a future project developing pelvic fixations could be very appealing. If a successful FEA model was developed, the proposed fixations could be tested using the FEA software and for the final assessment of their result an experiment on them could make a highly valuable study.

References

[1] G. Bergmann, G. Deuretzbacher, M. Heller, F. Graichen, a Rohlmann, J. Strauss, and G. . Duda, “Hip contact forces and gait patterns from routine activities,” *J. Biomech.*, vol. 34, no. 7, pp. 859–871, Jul. 2001.

[2] P. A. O’Neill, J. Riina, S. Sclafani, and P. Tornetta, “Angiographic findings in pelvic fractures.,” *Clin. Orthop. Relat. Res.*, no. 329, pp. 60–7, Aug. 1996.

[3] C. Moreno, E. E. Moore, A. Rosenberger, and H. C. Cleveland, “Hemorrhage associated with major pelvic fracture: a multispecialty challenge.,” *J. Trauma*, vol. 26, no. 11, pp. 987–94, Nov. 1986.

[4] Dwight Thomas and G. A. Piersol, *Human anatomy : including structure and development and practical considerations / by Thomas Dwight ... [et al.] ; edited by George A. Piersol.* Philadelphia :Lippincott,.

[5] K. Moore, *Clinically oriented anatomy*. 2014.

[6] R. Drake, A. W. Vogl, and A. W. M. Mitchell, *Gray’s Anatomy for Students*. Elsevier Health Sciences, 2014.

[7] “Pelvis Fractures,” *American Academy of Orthopaedic Surgeons (AAOS)*, 2007. [Online]. Available: http://orthoinfo.aaos.org/topic.cfm?topic=a00223. [Accessed: 27-Oct-2014].

[8] N. Palastanga, D. Field, and R. Soames, *Anatomy and human movement : structure and function*. 1998.

[9] H. A. C. Jacob, A. H. Huggler, C. Dietschi, and A. Schreiber, “Mechanical function of subchondral bone as experimentally determined on the acetabulum of the human pelvis,” *J. Biomech.*, vol. 9, no. 10, pp. 625–627, Jan. 1976.

[10] A. Cappozzo, F. Catani, U. Della Croce, and A. Leardini, “Position and orientation in space of bones during movement: anatomical frame definition and determination,” *Clin. Biomech.*, vol. 10, no. 4, pp. 171–178, Jun. 1995.

[11] U. Della Croce, A. Cappozzo, and D. C. Kerrigan, “Pelvis and lower limb anatomical landmark calibration precision and its propagation to bone geometry and joint angles,” *Med. Biol. Eng. Comput.*, vol. 37, no. 2, pp. 155–161, Mar. 1999.

[12] A. S. O. Leung, L. M. Gordon, T. Skrinskas, T. Szwedowski, and C. M. Whyne, “Effects of bone density alterations on strain patterns in the pelvis: application of a finite element model.,” *Proc. Inst. Mech. Eng. H.*, vol. 223, no. 8, pp. 965–979, 2009.

[13] A. Salavati, V. Shah, Z. J. Wang, B. M. Yeh, N. G. Costouros, and F. V Coakley, "F-18 FDG PET/CT findings in postradiation pelvic insufficiency fracture.," *Clin. Imaging*, vol. 35, no. 2, pp. 139–42, 2011.

[14] A. Donovan, M. E. Schweitzer, M. Rafii, and A. Lax, "Radiological features of superomedial iliac insufficiency fractures: A possible mimicker of metastatic disease," *Skeletal Radiol.*, vol. 38, no. 1, pp. 43–49, 2009.

[15] G. F. Pennal, M. Tile, J. P. Waddell, and H. Garside, "Pelvic disruption: assessment and classification.," *Clin. Orthop. Relat. Res.*, no. 151, pp. 12–21, Sep. 1980.

[16] A. R. Burgess, B. J. Eastridge, and J. W. Young, "Pelvic ring disruptions: effective classification system and treatment protocols.," vol. 30, no. 7, pp. 848–56, 1990.

[17] "A Reference Guide to Osteoporosis Reimbursement Policy for Healthcare Professionals," 2012.

[18] "Osteoporosis," *Public Health Agency of Canada*, 2009. [Online]. Available: http://www.phac-aspc.gc.ca/cd-mc/osteoporosis-osteoporose-eng.php.

[19] "Osteoporosis Facts & Statistics," *Osteoporosis Canada*, 2014. [Online]. Available: http://www.osteoporosis.ca/osteoporosis-and-you/osteoporosis-facts-and-statistics/.

[20] R. Raman, C. S. Roberts, H.-C. Pape, and P. V Giannoudis, "Implant retention and removal after internal fixation of the symphysis pubis.," *Injury*, vol. 36, no. 7, pp. 827–31, Jul. 2005.

[21] Transport Canada, "Canadian Motor Vehicle Traffic Collision Statistics: 2012," Jul. 2014.

[22] P. C. Dischinger, B. Cushing, and T. Kerns, "Injury patterns associated with direction of impact: drivers admitted to trauma centers," *J Trauma*, vol. 35, no. 3, pp. 454–8, 1993.

[23] R. Mehin, B. Jones, Q. Zhu, and H. Broekhuyse, "A biomechanical study of conventional acetabular internal fracture fixation versus locking plate fixation," *Can J Surg*, vol. 52, no. 3, pp. 221–228, 2009.

[24] V. B. Shim, J. Böshme, P. Vaitl, C. Josten, and I. a Anderson, "An efficient and accurate prediction of the stability of percutaneous fixation of acetabular fractures with finite element simulation.," *J. Biomech. Eng.*, vol. 133, no. 9, p. 094501, Sep. 2011.

[25] X.-W. Liu, S.-G. Xu, Y.-T. Zhang, and C.-C. Zhang, "Biomechanical Study of Acetabular Tridimensional Memoryalloy Fixation System," *J. Mater. Eng. Perform.*, vol. 20, no. 4–5, pp. 671–678, Jan. 2011.

[26] J. Böhme, V. Shim, a Höch, M. Mütze, C. Müller, and C. Josten, "Clinical implementation of finite element models in pelvic ring surgery for prediction of implant behavior: a case report.," *Clin. Biomech. (Bristol, Avon)*, vol. 27, no. 9, pp. 872–8, Nov. 2012.

[27] T. Bodzay, I. Flóris, and K. Váradi, "Comparison of stability in the operative treatment of pelvic injuries in a finite element model.," *Arch. Orthop. Trauma Surg.*, vol. 131, no. 10, pp. 1427–33, Oct. 2011.

[28] J. M. García, M. Doblaré, B. Seral, F. Seral, D. Palanca, and L. Gracia, "Three-Dimensional Finite Element Analysis of Several Internal and External Pelvis Fixations," *J. Biomech. Eng.*, vol. 122, no. 5, pp. 516–522, May 2000.

[29] V. Shim, J. Böhme, P. Vaitl, S. Klima, C. Josten, and I. Anderson, "Finite element analysis of acetabular fractures--development and validation with a synthetic pelvis.," *J. Biomech.*, vol. 43, no. 8, pp. 1635–9, May 2010.

[30] J. P. Clements, N. Moriaty, T. J. S. Chesser, a J. Ward, and J. L. Cunningham, "Determination of pelvic ring stability: a new technique using a composite hemi-pelvis," *Proc. Inst. Mech. Eng. Part H J. Eng. Med.*, vol. 222, no. 5, pp. 611–616, May 2008.

[31] H. C. Sagi, N. R. Ordway, and T. DiPasquale, "Biomechanical analysis of fixation for vertically unstable sacroiliac dislocations with iliosacral screws and symphyseal plating.," *J. Orthop. Trauma*, vol. 18, no. 3, pp. 138–143, 2004.

[32] T. Dawei, L. Na, L. Jun, J. Wei, and C. Lin, "A novel fixation system for sacroiliac dislocation fracture: internal fixation system design and biomechanics analysis.," *Clin. Biomech. (Bristol, Avon)*, vol. 28, no. 2, pp. 129–33, Feb. 2013.

[33] V. B. Shim, R. P. Pitto, R. M. Streicher, P. J. Hunter, and I. a Anderson, "Development and validation of patient-specific finite element models of the hemipelvis generated from a sparse CT data set.," *J. Biomech. Eng.*, vol. 130, no. 5, p. 051010, Oct. 2008.

[34] R. S. Salzar, D. Genovese, C. R. Bass, J. R. Bolton, H. Guillemot, a. M. Damon, and J. R. Crandall, "Load path distribution within the pelvic structure under lateral loading," *Int. J. Crashworthiness*, vol. 14, no. 1, pp. 99–110, Feb. 2009.

[35] P. T. Simonian, M. L. C. R. Jr, and A. F. Tencer, "The unstable iliac fracture : a biomechanical evaluation of internal fixation," vol. 28, no. 7, pp. 469–475, 1997.

[36] A. E. Anderson, C. L. Peters, B. D. Tuttle, and J. a. Weiss, "Subject-Specific Finite Element Model of the Pelvis: Development, Validation and Sensitivity Studies," *J. Biomech. Eng.*, vol. 127, no. 3, p. 364, 2005.

[37] E. Varga, T. Hearn, J. Powell, and M. Tile, "Effects of method of internal fixation of symphyseal disruptions on stability of the pelvic ring," *Injury*, vol. 26, no. 2, pp. 75–80, 1995.

[38] A. S. Dickinson, A. C. Taylor, and M. Browne, “The influence of acetabular cup material on pelvis cortex surface strains, measured using digital image correlation,” *J. Biomech.*, vol. 45, no. 4, pp. 719–723, Feb. 2012.

[39] R. Ghosh, S. Gupta, A. Dickinson, and M. Browne, “Experimental validation of finite element models of intact and implanted composite hemi-pelvises using digital image correlation,” *J. Biomech. Eng.*, vol. 134, no. August 2012, pp. 1–9, 2012.

[40] Z. Li, J.-E. Kim, J. S. Davidson, B. S. Etheridge, J. E. Alonso, and A. W. Eberhardt, “Biomechanical response of the pubic symphysis in lateral pelvic impacts: a finite element study.,” *J. Biomech.*, vol. 40, no. 12, pp. 2758–66, Jan. 2007.

[41] R. Philippot, J. Wegrzyn, F. Farizon, and M. H. Fessy, “Pelvic balance in sagittal and Lewinnek reference planes in the standing, supine and sitting positions.,” *Orthop. Traumatol. Surg. Res.*, vol. 95, no. 1, pp. 70–6, Feb. 2009.

[42] M. Dalstra, “Development and Validation of a Three-Dimensional Finite Element Model of the Pelvic Bone,” *J. Biomech. Eng.*, vol. 117, no. 3, p. 272, Oct. 2007.

[43] M. Sutton, H. W. Schreier, and J.-J. Orteu, *Image correlation for shape, motion and deformation measurements basic concepts, theory and applications*. New York : Springer, 2009.

[44] T. C. Chu, W. F. Ranson, and M. A. Sutton, “Applications of digital-image-correlation techniques to experimental mechanics,” *Exp. Mech.*, vol. 25, no. 3, pp. 232–244, Sep. 1985.

[45] H. A. Bruck, S. R. McNeill, M. A. Sutton, and W. H. Peters, “Digital image correlation using Newton-Raphson method of partial differential correction,” *Exp. Mech.*, vol. 29, no. 3, pp. 261–267, Sep. 1989.

[46] L. Petersson, I. Kvien, and K. Oksman, “Structure and thermal properties of poly(lactic acid)/cellulose whiskers nanocomposite materials,” *Compos. Sci. Technol.*, vol. 67, no. 11–12, pp. 2535–2544, Sep. 2007.

[47] W. H. Peters and W. F. Ranson, “Digital Imaging Techniques In Experimental Stress Analysis,” *Opt. Eng.*, vol. 21, no. 3, p. 213427, Jun. 1982.

[48] A. Sharir, M. M. Barak, and R. Shahar, “Whole bone mechanics and mechanical testing.,” *Vet. J.*, vol. 177, no. 1, pp. 8–17, Jul. 2008.

[49] S. Gupta, F. C. T. van der Helm, J. C. Sterk, F. van Keulen, and B. L. Kaptein, “Development and experimental validation of a three-dimensional finite element model of the human scapula,” *Proc. Inst. Mech. Eng. Part H J. Eng. Med.*, vol. 218, no. 2, pp. 127–142, Jan. 2004.

[50] E. Schileo, F. Taddei, A. Malandrino, L. Cristofolini, and M. Viceconti, “Subject-specific finite element models can accurately predict strain levels in long bones.,” *J. Biomech.*, vol. 40, no. 13, pp. 2982–9, Jan. 2007.

[51] F. Taddei, L. Cristofolini, S. Martelli, H. S. Gill, and M. Viceconti, “Subject-specific finite element models of long bones: An in vitro evaluation of the overall accuracy.,” *J. Biomech.*, vol. 39, no. 13, pp. 2457–67, Jan. 2006.

[52] A. S. Dickinson, a C. Taylor, H. Ozturk, and M. Browne, “Experimental validation of a finite element model of the proximal femur using digital image correlation and a composite bone model.,” *J. Biomech. Eng.*, vol. 133, no. 1, p. 014504, Jan. 2011.

[53] D. P. Nicolella, A. E. Nicholls, J. Lankford, and D. T. Davy, “Machine vision photogrammetry: a technique for measurement of microstructural strain in cortical bone,” *J. Biomech.*, vol. 34, no. 1, pp. 135–139, Jan. 2001.

[54] M. S. Thompson, H. Schell, J. Lienau, and G. N. Duda, “Digital image correlation: a technique for determining local mechanical conditions within early bone callus.,” *Med. Eng. Phys.*, vol. 29, no. 7, pp. 820–3, Sep. 2007.

[55] S. Amin Yavari, J. van der Stok, H. Weinans, and A. A. Zadpoor, “Full-field strain measurement and fracture analysis of rat femora in compression test.,” *J. Biomech.*, vol. 46, no. 7, pp. 1282–92, Apr. 2013.

[56] H. Yamaguchi, H. Kikugawa, T. Asaka, H. Kasuya, and M. Kuninori, “Measurement of Cortical Bone Strain Distribution by Image Correlation Techniques and from Fracture Toughness,” *Mater. Trans.*, vol. 52, no. 5, pp. 1026–1032, 2011.

[57] L. T. Sojic, A. Milic Lemic, I. Tanasic, N. Mitrovic, M. Milosevic, and A. Petrovic, “Compressive strains and displacement in a partially dentate lower jaw rehabilitated with two different treatment modalities.,” *Gerodontology*, vol. 29, no. 2, pp. e851–7, Jun. 2012.

[58] I. Tanasic, L. Tihacek-Sojić, A. M. Lemic, M. Djuri, N. Mitrovi, M. Milo, and A. Sedmak, “Optical Aspect of Deformation Analysis in the Bone-Denture Complex,” vol. 36, pp. 173–178, 2012.

[59] G. Benecke, M. Kerschnitzki, P. Fratzl, and H. S. Gupta, “Digital image correlation shows localized deformation bands in inelastic loading of fibrolamellar bone,” *J. Mater. Res.*, vol. 24, no. 02, pp. 421–429, Jan. 2011.

[60] J. Gonzalez and W. G. Knauss, “Strain inhomogeneity and discontinuous crack growth in a particulate composite,” *J. Mech. Phys. Solids*, vol. 46, no. 10, pp. 1981–1995, Oct. 1998.

[61] T. A. Berfield, J. K. Patel, R. G. Shimmin, P. V. Braun, J. Lambros, and N. R. Sottos, “Micro- and Nanoscale Deformation Measurement of Surface and Internal Planes via Digital Image Correlation,” *Exp. Mech.*, vol. 47, no. 1, pp. 51–62, Jan. 2007.

[62] G. Vendnroux, N. Schmidt, and W. G. Knauss, "Submicron deformation field measurements: Part 3. Demonstration of deformation determinations," *Exp. Mech.*, vol. 38, no. 3, pp. 154–160, Sep. 1998.

[63] I. Chasiotis and W. G. Knauss, "A new microtensile tester for the study of MEMS materials with the aid of atomic force microscopy," *Exp. Mech.*, vol. 42, no. 1, pp. 51–57, Mar. 2002.

[64] J. D. Craik, C. H. L. Laffer, S. W. Richards, S. P. Walsh, and S. L. Evans, "Distal humerus cortical strains following total elbow arthroplasty," *J. Eng. Med.*, vol. 227, no. 2, pp. 120–128, 2013.

[65] A. S. Dickinson, A. C. Taylor, and M. Browne, "Performance of the resurfaced hip. Part 1: the influence of the prosthesis size and positioning on the remodelling and fracture of the femoral neck," *Proc. Inst. Mech. Eng. Part H J. Eng. Med.*, vol. 224, no. 3, pp. 427–439, Mar. 2010.

[66] Z. Gao and J. P. Desai, "Estimating zero-strain states of very soft tissue under gravity loading using digital image correlation.," *Med. Image Anal.*, vol. 14, no. 2, pp. 126–37, Apr. 2010.

[67] K. M. Moerman, C. a Holt, S. L. Evans, and C. K. Simms, "Digital image correlation and finite element modelling as a method to determine mechanical properties of human soft tissue in vivo.," *J. Biomech.*, vol. 42, no. 8, pp. 1150–3, May 2009.

[68] L. Horny, H. Chlup, R. Zitny, T. Vonavkova, J. Vesely, and P. Lanzer, "Ex Vivo Coronary Stent Implantation Evaluated with Digital Image Correlation," *Exp. Mech.*, vol. 52, pp. 1555–1558, 2012.

[69] K. Genovese, Y.-U. Lee, a Y. Lee, and J. D. Humphrey, "An improved panoramic digital image correlation method for vascular strain analysis and material characterization.," *J. Mech. Behav. Biomed. Mater.*, vol. 27, pp. 132–42, Nov. 2013.

[70] K. Genovese, Y. U. Lee, and J. D. Humphrey, "Novel optical system for in vitro quantification of full surface strain fields in small arteries: II. Correction for refraction and illustrative results.," *Comput. Methods Biomech. Biomed. Engin.*, vol. 14, no. 3, pp. 227–37, Mar. 2011.

[71] J. L. Dressler, "Development of the Multi-Axis Biomechanical Testing Apparatus to Experimentally Measure In-Vitro Stresses on Knee Cortex under Gait Level Loads," University of Alberta, 2008.

[72] C. Tai, C. Lin, H. Wang, D. Lee, and P. Hsieh, "Stress Distribution of a Modified Periacetabular Osteotomy for Treatment of Dysplastic Acetabulum," vol. 31, no. 1, pp. 53–58, 2010.

[73] S. Majumder, A. Roychowdhury, and S. Pal, “Dynamic response of the pelvis under side impact load – a three-dimensional finite element approach,” *Int. J. Crashworthiness*, vol. 9, no. 1, pp. 89–103, Jan. 2004.

[74] E. A. Zimmermann, M. E. Launey, H. D. Barth, and R. O. Ritchie, “Mixed-mode fracture of human cortical bone.,” *Biomaterials*, vol. 30, no. 29, pp. 5877–84, Oct. 2009.

[75] E. A. Zimmermann, M. E. Launey, and R. O. Ritchie, “The significance of crack-resistance curves to the mixed-mode fracture toughness of human cortical bone.,” *Biomaterials*, vol. 31, no. 20, pp. 5297–305, Jul. 2010.

[76] C. Livermore and J. Voldman, “6.777J/2.751J Material Properties Database,” *MIT*. [Online]. Available: http://web.mit.edu/6.777/www/. [Accessed: 01-Jan-2001].

[77] N. J. Lawson and J. Wu, “Three-dimensional particle image velocimetry: experimental error analysis of a digital angular stereoscopic system,” *Meas. Sci. Technol.*, vol. 8, no. 12, pp. 1455–1464, Dec. 1997.

[78] J. D. Currey, “Physical characteristics affecting the tensile failure properties of compact bone,” *J. Biomech.*, vol. 23, no. 8, pp. 837–844, Jan. 1990.

[79] J. D. Currey, *Bones: Structure and Mechanics*. Princeton University Press, 1984.

[80] J. D. Currey, “Anelasticity in Bone and Echinoderm Skeletons,” *J. Exp. Biol.*, vol. 43, pp. 279–292, 1965.

[81] R. S. Lakes, J. L. Katz, and S. S. Sternstein, “Viscoelastic properties of wet cortical bone—I. Torsional and biaxial studies,” *J. Biomech.*, vol. 12, no. 9, pp. 657–678, Jan. 1979.

Appendix

I. The Matlab Code of the Gap Program:

The can be found in the file: “gap.m”.

```
function  answer = gap(xx1,yy1,xx2,yy2,imagenum,address)


point1 = [xx1,yy1];

point2 = [xx2,yy2];


address = address (1:end-7);


tmp = 1000 + imagenum;

tmp = num2str (tmp);

tmp = tmp (2:end);

address = [address tmp '.vc7'];


vecFile    = address;

v = load_3DVC7(vecFile,1);


% dimension of the mainmatrix

[xnum,ynum] = size (v.vx);


%finding the indexes of the of the desired coordinates

for i=1:xnum;

   if (v.x(1,i) - point1(1)) > 0

       if (v.x(1,i) - point1(1)) < (point1(1) - v.x(1,i-1))

           xnp1 = i;
```

```
        else
            xnp1 = i-1;
        break;
        end
    end
 end

for i=1:ynum;
    if (v.y(1,i) - point1(2)) > 0
        if (v.y(1,i) - point1(2)) < (point1(2) - v.y(1,i-1))
            ynp1 = i;
        else
            ynp1 = i-1;
        end
        break;
    end
  end

for i=1:xnum;
    if (v.x(1,i) - point2(1)) > 0
        if (v.x(1,i) - point2(1)) < (point2(1) - v.x(1,i-1))
            xnp2 = i;
        else
            xnp2 = i-1;
        end
        break;
    end
end
```

```
for i=1:ynum;
   if (v.y(1,i) - point2(2)) > 0
      if (v.y(1,i) - point2(2)) < (point2(2) - v.y(1,i-1))
         ynp2 = i;
      else
         ynp2 = i-1;
      end
      break;
   end
end

real_x_point_1 = v.x(1,xnp1);
real_y_point_1 = v.y(1,ynp1);
real_z_point_1 = v.z(1,ynp1);

real_x_point_2 = v.x(1,xnp2);
real_y_point_2 = v.y(1,ynp2);
real_z_point_2 = v.z(1,ynp2);

 dx1 = v.vx (xnp1,ynp1);
 dx2 = v.vx (xnp2,ynp2);

 dy1 = v.vy (xnp1,ynp1);
 dy2 = v.vy (xnp2,ynp2);

 dz1 = v.vw (xnp1,ynum-ynp1);
 dz2 = v.vw (xnp2,ynum-ynp2);
```

```
x1 = real_x_point_1 + dx1;
x2 = real_x_point_2 + dx2;

y1 = real_y_point_1 - dy1;
y2 = real_y_point_2 - dy2;

z1 = real_z_point_1 + dz1;
z2 = real_z_point_2 + dz2;

answer = sqrt ( (x1-x2)^2 + (y1-y2)^2 + (z1-z2)^2 );
```

II. The Matlab Code of the Gap Mode Program:

The code can be found in the file: “gapmode.m”.

```
% ----------------------------------------------------------------------
%   this finction gets 5 points: an origin, a pint on the s axis, a point on
%   the y axis, and two arbitray points for which we are to determin how they
%   move realatively to the our coordinate system
% ----------------------------------------------------------------------

function ans = gapmode (imagenum)
%(ox,oy,xx,xy,yx,yy,xx1,yy1,xx2,yy2,imagenum,address)%

[point_o] = [0,10];
[point_x] = [10,25];
[point_y] = [0,12];
[point1] = [8,9];
[point2] = [12,19];
%imagenum = 13;
address = 'C:\Users\Hamed\Desktop\4\B00002.vc7';

% point_o = [ox,oy];
% point_x = [xx,xy];
% point_y = [yx,yy];

address = address (1:end-7);

tmp = 1000 + imagenum;
tmp = num2str (tmp);
tmp = tmp (2:end);
address = [address tmp '.vc7'];

vecFile    = address;
v = load_3DVC7(vecFile,1);

% dimension of the mainmatrix
[xnum,ynum] = size (v.vx);

%finding the indexes of the desired coordinates
%for the point o (the origin)

for i=1:xnum;
    if (v.x(1,i) - point_o(1)) > 0
        if (v.x(1,i) - point_o(1)) < (point_o(1) - v.x(1,i-1))
            xnp_o = i;
        else
            xnp_o = i-1;
        break;
        end
    end
end

for i=1:ynum;
    if (v.y(1,i) - point_o(2)) > 0
        if (v.y(1,i) - point_o(2)) < (point_o(2) - v.y(1,i-1))
            ynp_o = i;
        else
```

```
            ynp_o = i-1;
        end
        break;
    end

end

%for the point x (on the x axis)
for i=1:xnum;
    if (v.x(1,i) - point_x(1)) > 0
        if (v.x(1,i) - point_x(1)) < (point_x(1) - v.x(1,i-1))
            xnp_x = i;
        else
            xnp_x = i-1;
        break;
        end
    end
end

for i=1:ynum;
    if (v.y(1,i) - point_x(2)) > 0
        if (v.y(1,i) - point_x(2)) < (point_x(2) - v.y(1,i-1))
            ynp_x = i;
        else
            ynp_x = i-1;
        end
        break;
    end
```

```
end


%for the point y (point on the y axis)
for i=1:xnum;
    if (v.x(1,i) - point_y(1)) > 0
        if (v.x(1,i) - point_y(1)) < (point_y(1) - v.x(1,i-1))
            xnp_y = i;
        else
            xnp_y = i-1;
        break;
        end
    end
end


for i=1:ynum;
    if (v.y(1,i) - point_y(2)) > 0
        if (v.y(1,i) - point_y(2)) < (point_y(2) - v.y(1,i-1))
            ynp_y = i;
        else
            ynp_y = i-1;
        end
        break;
    end
end


% finding the existing coordinates of the nearest points to the selected
% points
```

```
real_x_point_o = v.x(1,xnp_o);
real_y_point_o = v.y(1,ynp_o);
real_z_point_o = v.z(1,ynp_o);

real_x_point_x = v.x(1,xnp_x);
real_y_point_x = v.y(1,ynp_x);
real_z_point_x = v.z(1,ynp_x);

real_x_point_y = v.x(1,xnp_y);
real_y_point_y = v.y(1,ynp_y);
real_z_point_y = v.z(1,ynp_y);

 dx_o = v.vx (xnp_o,ynp_o);
 dx_x = v.vx (xnp_x,ynp_x);
 dx_y = v.vx (xnp_y,ynp_y);

 dy_o = v.vy (xnp_o,ynp_o);
 dy_x = v.vy (xnp_x,ynp_x);
 dy_y = v.vy (xnp_y,ynp_y);

 dz_o = v.vw (xnp_o,ynum-ynp_o);
 dz_x = v.vw (xnp_x,ynum-ynp_x);
 dz_y = v.vw (xnp_y,ynum-ynp_y);

 current_ox = real_x_point_o + dx_o;
 current_xx = real_x_point_x + dx_x;
 current_yx = real_x_point_y + dx_y;
 current_oy = real_y_point_o - dy_o;

 current_xy = real_y_point_x - dy_x;

 current_yy = real_y_point_y - dy_y;


 current_oz = real_z_point_o + dz_o;

 current_xz = real_z_point_x + dz_x;

 current_yz = real_z_point_y + dz_y;


%finding the unit vector along the x direction

unitvec_x = [current_xx - current_ox , current_xy - current_oy , current_xz - current_oz] ;

unitvec_x = unitvec_x / sqrt (unitvec_x * unitvec_x');


a_vector_y = [current_yx - current_ox , current_yy - current_oy , current_yz - current_oz] ;

%%unitvec_y = unitvec_y / sqrt ((real_x_point_y - real_x_point_o)^2+(real_y_point_y - real_y_point_o)^2+(real_z_point_y - real_z_point_o)^2);


unitvec_z = cross (unitvec_x , a_vector_y);

unitvec_z = unitvec_z / sqrt(unitvec_z * unitvec_z');


unitvec_y = cross (unitvec_z,unitvec_x);


%--------------------------------------------------------------

%finding the displacement and location of the two points

%--------------------------------------------------------------

% point1 = [xx1,yy1];

% point2 = [xx2,yy2];

%finding the indexes of the of the desired coordinates


for i=1:xnum;
```

```
    if (v.x(1,i) - point1(1)) > 0
        if (v.x(1,i) - point1(1)) < (point1(1) - v.x(1,i-1))
            xnp1 = i;
        else
            xnp1 = i-1;
        break;
        end
    end
end

for i=1:ynum;
    if (v.y(1,i) - point1(2)) > 0
        if (v.y(1,i) - point1(2)) < (point1(2) - v.y(1,i-1))
            ynp1 = i;
        else
            ynp1 = i-1;
        end
        break;
    end
end

for i=1:xnum;
    if (v.x(1,i) - point2(1)) > 0
        if (v.x(1,i) - point2(1)) < (point2(1) - v.x(1,i-1))
            xnp2 = i;
        else
            xnp2 = i-1;
        end
        break;
    end
end

for i=1:ynum;
    if (v.y(1,i) - point2(2)) > 0
        if (v.y(1,i) - point2(2)) < (point2(2) - v.y(1,i-1))
            ynp2 = i;
        else
            ynp2 = i-1;
        end
        break;
    end
end

  dx1 = v.vx (xnp1,ynp1);
  dx2 = v.vx (xnp2,ynp2);

  dy1 = v.vy (xnp1,ynp1);
  dy2 = v.vy (xnp2,ynp2);

  dz1 = v.vw (xnp1,ynum-ynp1);
  dz2 = v.vw (xnp2,ynum-ynp2);

  % finding the gap from the first image until the image number 'n'
  disp_x = dx2 - dx1;
  disp_y = dy2 - dy1;
  disp_z = dz2 - dz1;
```

```
disp = [disp_x , disp_y , disp_z];

% trandformation to the new coordinate system with the o',x',y'
X = abs(disp * unitvec_x');
Y = abs(disp * unitvec_y');
Z = abs(disp * unitvec_z');
ans = [X,Y,Z];
```

III. CVI Program

In this section the developed CVI code is presented. A CVI project requires several files:

1. The “.h” files or the header files in which the headers and the main parameters are defined.
2. The “.uir” files or the user interface requirements in which the graphical user interface (GUI) of the program is designed. The images in the chapter 2 show the panels and handles of the GUIs.
3. The “.c” or the c code files in which the main commands and code of the software is written.

The CVI code developed in the “modify___MABTA_Ctrl.c” file can be found in this section.

```
// MABTA_Ctrl
//
// A MABTA_Ctrl log and disply program          :          to communicate with the MABTA 6 DOF Load Cell and Motor Drive

#include "toolbox.h"
#include <string.h>
#include <gpib.h>
#include <formatio.h>
#include <rs232.h>
#include <utility.h>
#include <ansi_c.h>
#include <analysis.h>
#include <cvirte.h>
#include <userint.h>
#include <ctype.h>
#include <MABTA_Ctrl_Declare.h>      // here it knows to look in these files for the declaired variables
#include <MABTA_Ctrl.h>


//------------------------------------------------
// QB Load Cell COMMANDS :
//------------------------------------------------
char LC_start[]= "Q";
char LC_stop[]= "R";


//------------------------------------------------
// QB Cammera Commands :
//------------------------------------------------
/*char CAM_start[]= "S";
char CAM_max[]= "M";
char CAM_stop[]= "E";*/


//*******************************************************
// main                                  : MCP main function
//*******************************************************
int main (int argc, char *argv[])
{
        if (InitCVIRTE (0, argv, 0) == 0)
                return -1;/* out of memory */
        if ((panelHandle = LoadPanel (0, "MABTA_Ctrl.uir", PANEL)) < 0)
                return -1;

        // Set the panel variables
                g_Handle            = LoadPanel (0, "MABTA_Ctrl.uir", G_SETUP);
                com_Handle          = LoadPanel (0, "MABTA_Ctrl.uir", COM);
                a_Handle            = LoadPanel (0, "MABTA_Ctrl.uir", ABOUT);
                s_Handle            = LoadPanel (0, "MABTA_Ctrl.uir", SAVE_Con);
                v_corr_Handle       = LoadPanel (0, "MABTA_Ctrl.uir", V_CORR);
                HN_clbr_Handle      = LoadPanel (0, "MABTA_Ctrl.uir", PANEL_CLBR);

                HN_MGHandle         = LoadPanel (0, "MABTA_Ctrl.uir", GPanel);
                HN_FEHandle = LoadPanel (0,"MABTA_Ctrl.uir", FILE_ERROR);



                // Set tab panels

                for (i=0;i<6;i++)
                {
                        GetPanelHandleFromTabPage (panelHandle, PANEL_TAB, i, &HN_TABHandle[i]);
                }



        DisplayPanel (panelHandle);
                DSN_Init();
                DSN_Init2();
        SetSleepPolicy (VAL_SLEEP_MORE);
        // Maximuise the panel
        //SetPanelAttribute (panelHandle, ATTR_WINDOW_ZOOM, VAL_MAXIMIZE);
        RunUserInterface ();

        Error:
        QB_Kill_Motor();
        DiscardPanel (panelHandle);
        CloseCVIRTE ();
        return 0;
```

```
}

//*********************************************************
// ExitCallback                    : Exit menu
//*********************************************************
void CVICALLBACK ExitCallback (int menuBar, int menuItem, void *callbackData, int panel)

{
                              DisplayPanel (s_Handle);
}

//*********************************************************
// DSN_MainPanelQuit: Main panel quit function
//*********************************************************
void DSN_MainPanelQuit(void)
{int i;

          QB_Kill_Motor();                                                                              //Stop
and disable the motor drive

          QuitUserInterface (0);


return;
}
//*********************************************************
// Save_MCP_Config: Save the current configuration on exit
//*********************************************************
int CVICALLBACK Save_MCP_Config (int panel, int control, int event,
                    void *callbackData, int eventData1, int eventData2)
{
          switch (event)
                    {
                    case EVENT_COMMIT:
                    switch (control)
                              {
                              case SAVE_Con_SAVE_YES:
                                        DSN_Save_Config(0);
                                        HidePanel (s_Handle);
                                        DSN_MainPanelQuit();
                                        break;
                              case SAVE_Con_SAVE_NO:
                                        HidePanel (s_Handle);
                                        DSN_MainPanelQuit();
                                        break;
                              case SAVE_Con_SAVE_CANCEL:
                                        HidePanel (s_Handle);
                                        break;
                              }
                              break;
                    case EVENT_RIGHT_CLICK:

                              break;
                    }
          return 0;
}
//*********************************************************
// SHOW AND CLOSE CALLBACKS FOR OTHER PANELS
//*********************************************************

//*********************************************************
// GRAPH SETUP
//*********************************************************
int CVICALLBACK SHOW_Graphs (int panel, int control, int event,
                    void *callbackData, int eventData1, int eventData2)
{
          switch (event)
                    {
                    case EVENT_COMMIT:

                              break;
                    case EVENT_RIGHT_CLICK:
                              DisplayPanel (g_Handle);
                              break;
                    }
          return 0;
}

int CVICALLBACK CLOSE_Graphs (int panel, int control, int event,
                    void *callbackData, int eventData1, int eventData2)
{
          switch (event)
                    {
                    case EVENT_COMMIT:
                              HidePanel (g_Handle);
                              break;
                    case EVENT_RIGHT_CLICK:

                              break;
                    }
          return 0;
}

//*******************************************************
// ABOUT
//*******************************************************
void CVICALLBACK SHOW_About (int menuBar, int menuItem, void *callbackData,
                    int panel)
{
 DisplayPanel (a_Handle);
}

int CVICALLBACK CLOSE_About (int panel, int control, int event,
                    void *callbackData, int eventData1, int eventData2)
{
          switch (event)
                    {
                    case EVENT_COMMIT:
                              HidePanel (a_Handle);
                              break;
                    case EVENT_RIGHT_CLICK:

                              break;
                    }
          return 0;
}
//*******************************************************
// COMMUNICATIONS SETUP
//*******************************************************
void CVICALLBACK SHOW_Com (int menuBar, int menuItem, void *callbackData,
                    int panel)
{
 DisplayPanel (com_Handle);
}

int CVICALLBACK CLOSE_Com (int panel, int control, int event,
                    void *callbackData, int eventData1, int eventData2)
{
          switch (event)
                    {
                    case EVENT_COMMIT:
                              HidePanel (com_Handle);
                              break;
                    case EVENT_RIGHT_CLICK:

                              break;
                    }
          return 0;
}

//*******************************************************
// V_CORR SETUP
//*******************************************************
void CVICALLBACK SHOW_V_Corr (int menuBar, int menuItem, void *callbackData,
                    int panel)
{
 DisplayPanel (v_corr_Handle);
}

int CVICALLBACK CLOSE_V_Corr (int panel, int control, int event,
                    void *callbackData, int eventData1, int eventData2)
{
          switch (event)
                    {
                    case EVENT_COMMIT:
                              HidePanel(v_corr_Handle);
                              DSN_Save_Vars();
                              break;
```

```
        case EVENT_RIGHT_CLICK:
            break;
        }
    return 0;
}

//********************************************************
// LOG_ON_OFF               : Check what functions to run
//********************************************************
int CVICALLBACK LOG_ON_OFF (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
        {
        case EVENT_COMMIT:
            GetCtrlVal(panelHandle, PANEL_LOG_COM1,  &Is_COM1);
            GetCtrlVal(panelHandle, PANEL_LOG_COM2,  &Is_COM2);
            GetCtrlVal(panelHandle, PANEL_CAM_COM,            &Is_COM3);
            GetCtrlVal(panelHandle, PANEL_LOG_Graph,  &Is_Graph);
            GetCtrlVal(HN_TABHandle[1], TAB_TEST_LOG_LOGtoFile,&Is_Log);
            //GetCtrlVal(HN_TABHandle, TAB_TEST_TEST_TYPE,     &test_type);
            break;
        case EVENT_RIGHT_CLICK:

            break;
        }
    return 0;
}
//********************************************************
// START_STOP               : Start/Stop control of Timer Loop
//                                  : Use a seperate thread
//********************************************************
int CVICALLBACK START_STOP (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{   int test, test2;
    int i,j;
    //int bytes_read;


    //------------------------------------------------
    // QB MOTOR DRIVE COMMANDS WITHOUT CARRIAGE RETURN
    //------------------------------------------------
    char M_enable[4] = "EN";
    char M_jog_command[6] = "MJOG";
    char M_stop[6] = "STOP";
    char M_disable[5] = "DIS";
    char M_op0[10] = "OPMODE 0";


    switch (event)
        {
        case EVENT_COMMIT:
            DSN_Save_Vars();
            GetCtrlVal(HN_TABHandle[1], TAB_TEST_START_STOP, &test2);
            if(!test2)
            {   //*********************************
                // Stop and Kill the DAQ thread
                //*********************************
                DAQquitflag = 0;
                QB_Kill_Motor();
        //Stop and disable the motor drive
                Delay(1);
                CmtReleaseThreadPoolFunctionID (DEFAULT_THREAD_POOL_HANDLE,
DAQthreadID);
                Delay(1);
                /*if(Is_COM3)
                    {
                        FlushInQ (LC_comport);
                        FlushOutQ(LC_comport);

                        COM3quitflag = 0;
                        Delay(1);
                        CmtReleaseThreadPoolFunctionID
(DEFAULT_THREAD_POOL_HANDLE, DAQthreadID);
                    }     */
                Delay(1);
                CmtDiscardLock (lockHandle);
```

```
                        //QB_Kill_Motor();
        //Stop and disable the motor drive

                        //---------------------------------------------
                        // Send a stop signal to the camera control program
                        if (Is_COM3)
                        {   FlushInQ (CAM_comport);
                                FlushOutQ(CAM_comport);
                                ComWrt(CAM_comport, "E\n", 2);
// Note: E used as camera stop command
                        }


                        //En_Panel_Ctrls(0);
                //Enable the control buttons
                }
                else
                {
                                //   Dim all buttons that should not be pushed while test is in progress
                  //En_Panel_Ctrls(1);

                //-----------------------------------------------------------
                // If Motor check box is selected            :              START MOVING THE MOTOR
                if(Is_COM1)
                {
                        //Flush Input and Output Queues
                        FlushInQ (M_comport);
                        FlushOutQ(M_comport);

                        //Enable the motor drive
                        strcat (M_enable,"\r");
                        ComWrt (M_comport, M_enable, strlen(M_enable));

                        //Put into Operation Mode 0 : digtal velocity
                        strcat(M_op0, "\r");
                        ComWrt (M_comport, M_op0, strlen(M_op0));


                        GetCtrlVal( HN_TABHandle[1], TAB_TEST_M_TEST_SPEED, &M_sjog );
//Get the input jog speed from the GUI

                        // Assemble the move command based on the type of test:
                        //SetCtrlVal(panelHandle, PANEL_test_type_display,test_type);
                        /*if (test_type==1)
                        // if it is a tension test
                        {       M_sent_len =  sprintf( M_jog, "J -%1.2f" , M_sjog);
//Assemble move command :       "J -(velocity)"
                                M_downflag=1;
                                // Document that stage is moving down
                        }
                        else if (test_type==0)
                        // if it is a compression test
                        {       M_sent_len =  sprintf( M_jog, "J %1.2f" , M_sjog);
//Assemble move command :       "J (velocity)"

                        } */

                        GetCtrlVal(HN_TABHandle[1], TAB_TEST_TEST_TYPE,  &test_type);

                        if (test_type == 0)
                        // if it is a compression test
                        {       M_sent_len =  sprintf( M_jog, "J %1.2f" , M_sjog);
//Assemble move command :       "J (velocity)"
                        }
                        else
                                        // if it is a tension test
                        {       M_sent_len =  sprintf( M_jog, "J -%1.2f" , M_sjog);
//Assemble move command :       "J -(velocity)"
                                M_downflag=1;
                                // Document that stage is moving down
                        }
                        //M_sent_len =  sprintf( M_jog, "J %1.2f" , M_sjog);
//Assemble move command :       "J (velocity)"

                        strcat (M_jog,"\r");
                //Add the Carriage Return

                        ComWrt (M_comport, M_jog, strlen(M_jog));
//Send jog command to controller
```

```
                                QB_M_Feedback(6);
                                        //Read the feedback from the motor drive

                                ProcessSystemEvents();

                        }
                        //-------------------------------------
                        // If Load cell check box is selected        :
                        if(Is_COM2)
                        {
                        }

                        //-------------------------------------
                        // If Camera check box is selected          :  send command to start taking pictures
                        if(Is_COM3)
                        {   FlushInQ (CAM_comport);
                                FlushOutQ(CAM_comport);
                                ComWrt(CAM_comport,"S\n",2);
        //  Note: S used as camera start command
                        }

                        //-----------------------------------------------------
                        // If Graph box checked          :              initialize arrays with zeroes
                        if(Is_Graph)
                        {
                                 QB_Clear_Arrays();
                        }

                        DAQcmtStatus = CmtScheduleThreadPoolFunction (DEFAULT_THREAD_POOL_HANDLE,
                                                                                  DAQThreadFunction, NULL,
                                                                                  &DAQthreadID);
                         DAQquitflag = 1;
                        // Create a thread lock
                        CmtNewLock (NULL, 0, &lockHandle);

                        break;
                        }

                case EVENT_RIGHT_CLICK:

                        break;
                }
        return 0;
}
/******************************************************************************************/
/* DAQThreadFunction ():  A thread to run the tests in     */
/******************************************************************************************/
int CVICALLBACK DAQThreadFunction (void *functionData)
{       double T1,T2,Hz;
        short read_cnt;
        int bytes_read;

  double TEMP[10];
        int i, j, k = 0;
        int count = 0;

        double T0 = Timer ();                           //First call of timer : Get time at start of test

        SAMPLES = 0;

/*  Start a loop that will process events for this thread */
  while (DAQquitflag == 1)
  {     ++SAMPLES;
                        T1 = Timer ();                          //Get start time for one loop

                        ProcessSystemEvents ();
                        //--------------------------------------------
                        step[0] = (Timer () - T0);                       // Get Time relative to start of test
                        SetCtrlVal (HN_TABHandle[1], TAB_TEST_TIME_1, step[0]);
                        SetCtrlVal (HN_TABHandle[1], TAB_TEST_TIME_2, Timer ());

                        //--------------------------------------------
                        // Get The Position of the Traversing Stage
                        //--------------------------------------------
                        if(Is_COM1)
                        {
                                //Flush Input and Output Queues
                                //FlushInQ (M_comport);
                                FlushOutQ(M_comport);

                                M_read_data[0] = '\0';

                                ComWrt (M_comport, "PFB\r", 4);
                                //Send the position inquiry

                                //Expecting three things back :            "PFB" "position" "-->"

                                QB_M_Feedback(1);
                                        //Read "PFB"


                                M_inqlen = GetInQLen(M_comport);
                        //Get the input queu length
                                M_bytes_read = ComRdTerm(M_comport, M_position, M_inqlen, 10);
        //Read the position
                                CopyString (M_tbox_read_data, 0, M_position, 0, M_bytes_read);
                                SetCtrlVal(panelHandle, PANEL_M_FEEDBACK,M_tbox_read_data);



                                M_pos = atoi(M_position);
                                //Convert the position to an integer

                                QB_M_Feedback(1);
                                        //Read "-->"

                        }


                        ProcessSystemEvents ();

                        //----------------------------------------------
                        //  Get Data From Load Cell
                        //----------------------------------------------
                        if(Is_COM2)
                        {
                                //Flush Input and Output Queues
                                FlushInQ (LC_comport);
                                FlushOutQ(LC_comport);

                                QB_Read_LC();                           //Read Voltages From Load Cell

                                QB_Convert_FM();                        //Convert to Load Readings

                                //Flush Input and Output Queues
                                FlushInQ (LC_comport);
                                FlushOutQ(LC_comport);

                                ComWrt (LC_comport, LC_stop, StringLength(LC_stop));    //Stop flow delivery of load
cell data

                                ProcessSystemEvents ();
                        }

                        //----------------------------------------------
                        //  Check if Fz is greater than or equal to the the maximun load
                        //----------------------------------------------

                        if (Is_COM1 & Is_COM2)
                        {
                                GetCtrlVal(HN_TABHandle[1], TAB_TEST_MAX_LOAD, &max_load);

                        if (fabs(LC_FM_loads[2])>=max_load)
                        {
                                //Flush Input and Output Queues
                                FlushInQ (M_comport);
                                FlushOutQ(M_comport);

                                ComWrt (M_comport, "STOP\r", 5);
                        //Stop movement

                                QB_M_Feedback(1);
                                    //Read the feedback from the motor drive

                                //----------------------------------------------
                                // Send a signal to the camera control program
                                if (Is_COM3)
                                {        ComWrt(CAM_comport,"M\n",2);
                        // Note: M used as camera command to document max load image
                                }


                                //----------------------------------------------
                                // Send the traversing stage back in opposite direction
                                GetCtrlVal( HN_TABHandle[1], TAB_TEST_M_TEST_SPEED, &M_sjog );
        //Get the input jog speed from the GUI


                                // Assemble the move command based on the type of test:
                                if (test_type ==0)
                                // if it is a compression test
                                {
                                        M_sent_len =  sprintf( M_jog, "J -%1.2f" , M_sjog);
        //Assemble move command     :         "J -(velocity)"
                                        M_downflag=1;
                                        // Document that stage is moving down
                                }
                                else
                                                // if it is a tension test
                                {       M_sent_len =  sprintf( M_jog, "J %1.2f" , M_sjog);
        //Assemble move command     :         "J (velocity)"

                                }

                                //M_sent_len =  sprintf( M_jog, "J -%1.2f" , M_sjog);
        //Assemble move command     :         "J (velocity)"

                                strcat (M_jog,"\r");
                        //Add the Carriage Return

                                ComWrt (M_comport, M_jog, strlen(M_jog));
        //Send jog command to controller

                                QB_M_Feedback(2);
                                        //Read the feedback from the motor drive


                                //M_downflag = 1;
                                // Activate the next if statement
                                postmax_flag=1;
                                        // document that maximum load has been reached

                                ProcessSystemEvents();
                        }
                        //----------------------------------------------
                        //  Check physical limits and forces
                        //----------------------------------------------
                        if(M_downflag & postmax_flag)
                                //------------------------------------------------------------
                                //Stop Motor if load is 10% of maximum load
                                if (fabs(LC_FM_loads[2])<=0.1*max_load)
                                {
                                //      SetCtrlVal(panelHandle, PANEL_CHECK_LED, 1);
                                        ComWrt (M_comport, "STOP\r", 5);                               //Stop
movement

                                        QB_M_Feedback(2);
                        //Read the feedback from the motor drive

                                        M_downflag = 0;

                                        //----------------------------------------------
                                        // Send a stop signal to the camera control program
                                        if (Is_COM3)
                                        {       FlushInQ (CAM_comport);
                                          FlushOutQ(CAM_comport);
                                                ComWrt(CAM_comport, "E\n", 2);
        // Note: E used as camera stop command
                                        }
                                }

                        }
        //}
                        ProcessSystemEvents ();

                        //----------------------------------------------
                        DSN_UpdateScreen();                          //Update Values on the Main Panel

                        ProcessSystemEvents ();

                        //---------------------------------------------
                        QB_Update_Vars();                            //Update the arrays for the Log File

                        ProcessSystemEvents ();

                        //---------------------------------------------
                        if(Is_Log)
                        {
                                DSN_LogFile();                           // Log to File

                                ProcessSystemEvents ();
                        }
                        if(Is_Graph)
                        {

                                DSN_Graph();                             // Do Graphing
                                ProcessSystemEvents ();
                        }
                        //---------------------------------------------
                        DSN_SHIFT();                                 // SHIFT all arrays
                        ProcessSystemEvents ();
                        //---------------------------------------------
                         T2 = Timer ();                              //Get the time at the end of the loop
                        Hz = (T2-T1);
                        SetCtrlVal (HN_TABHandle[1], TAB_TEST_Sampling_Rate,1/Hz);    //Send the sampling
frequency to the GUI
                        //---------------------------------------------
        }
return 0;

}
//*******************************************************
// LC_READ_BUTTON
//*******************************************************
int CVICALLBACK LC_READ (int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)
{   int test;

        switch (event)
                        {
                        case EVENT_COMMIT:
                        DSN_Save_Vars();
                        GetCtrlVal(panelHandle, PANEL_LC_READ_BUTTON, &test);
                        if (!test)
                        {
                                LC_readflag = 0;

                                SetCtrlAttribute (panelHandle, PANEL_LC_ZERO_BUTTON, ATTR_DIMMED, 0);
//Enable the Zero button

                                SetCtrlAttribute (HN_TABHandle[1], TAB_TEST_START_STOP, ATTR_DIMMED, 0);
//Enable the test Start button
                        }
                        else
                        {
                                LC_readflag = 1;

                                SetCtrlAttribute (panelHandle, PANEL_LC_ZERO_BUTTON, ATTR_DIMMED, 1);
//Dim the Zero button

                                SetCtrlAttribute (HN_TABHandle[1], TAB_TEST_START_STOP, ATTR_DIMMED, 1);
//Dim the test Start button
                        }

                        while (LC_readflag == 1)
                        {
                                //Flush Input and Output Queues
                                FlushInQ (LC_comport);
                                FlushOutQ(LC_comport);

                                QB_Read_LC();                            //Read Voltages From Load Cell

                                QB_Convert_FM();                         //Convert to Load Readings

                                //Flush Input and Output Queues
```

```
                                        FlushInQ (LC_comport);
                                        FlushOutQ(LC_comport);

                                        ComWrt (LC_comport, LC_stop, StringLength(LC_stop));      //Stop delivery of load cell
data

                                        ProcessSystemEvents ();

                                        DSN_UpdateScreen();                                // Update Values on the Main Panel
                              }
                               break;
                              case EVENT_RIGHT_CLICK:

                              break;
                    }
          return 0;
}
//*******************************************************
// Zero Load Cell Callback     : Save current voltages as bias voltages
//*******************************************************
int CVICALLBACK LC_ZERO (int panel, int control, int event,
                    void *callbackData, int eventData1, int eventData2)
{
          switch (event)
          {
                    case EVENT_COMMIT:

                                        //Flush Input and Output Queues
                                        FlushInQ (LC_comport);
                                        FlushOutQ(LC_comport);


                                        SetCTSMode (LC_comport, LWRS_HWHANDSHAKE_OFF);
          //Turn off Hardware handshaking

                                        ComWrt (LC_comport, LC_start, StringLength(LC_start));          //Send the start
command

                                        //Read the FIRST byte
                                        bytes_read = ComRd (LC_comport, LC_first_byte, 1);

                                        //************* QB: Should I Put Something Here To Check If It's A
"U"???***********//

                                        //Read TWELVE bytes
                                        bytes_read = ComRd (LC_comport, LC_read_data, 12);

                                        len = StringLength(LC_read_data);

                                        for (t=0; t<len; t++)
                                        {
                                        QB_Print_Bin(LC_read_data[t]);                    //Send the first character to
printbin

                                        sprintf( LC_word, LC_onebyte);                    //Put the first character's
binary in word

                                        t++;
                                        QB_Print_Bin(LC_read_data[t]);                    //Send the second character
to printnin

                                        strcat(LC_word,LC_onebyte);                                 //Concatenate the
binary of the first and second characters into word

                                        QB_Get_Bias();

                                        }

                                        //Flush Input and Output Queues
                                        FlushInQ (LC_comport);
                                        FlushOutQ(LC_comport);

                                        ComWrt (LC_comport, LC_stop, StringLength(LC_stop));

                                        SetCtrlAttribute (panelHandle, PANEL_LC_ZERO_LED, ATTR_DIMMED, 0);
          //Turn on the Zeroed LED

                                        SetCtrlAttribute (HN_TABHandle[1], TAB_TEST_LOGFILE, ATTR_DIMMED, 0);
          // Enable the logfile button
```

```
                                    SetCtrlAttribute (HN_TABHandle[1], TAB_TEST_START_STOP, ATTR_DIMMED, 0);
        // Enable the START button

                                    ProcessSystemEvents ();

                            break;
        }
        return 0;
}
//********************************************************
// COM3ThreadFunction       : Here is another thread
//                                                  not in use in this version
//********************************************************
int CVICALLBACK COM3ThreadFunction (void *functionData)
{
  while (COM3quitflag == 1)
  {

        }
return 0;
}


//********************************************************
// DSN_UpdateScreen                 : Update all parameters on the
//                                                  : main panel
//********************************************************
void DSN_UpdateScreen(void)
{
        SetCtrlVal (HN_TABHandle[1], TAB_TEST_SAMPLES,     SAMPLES);

        if(Is_COM1)
        //Update the traversing stage position
                                    SetCtrlVal(panelHandle, PANEL_M_POSITION, M_pos);

        if(Is_COM2)
        //Update Load Values on Screen
                                    SetCtrlVal(panelHandle, PANEL_Fx, LC_FM_loads[0]);
                                    SetCtrlVal(panelHandle, PANEL_Fy, LC_FM_loads[1]);
                                    SetCtrlVal(panelHandle, PANEL_Fz, LC_FM_loads[2]);
                                    SetCtrlVal(panelHandle, PANEL_Mx, LC_FM_loads[3]);
                                    SetCtrlVal(panelHandle, PANEL_My, LC_FM_loads[4]);
                                    SetCtrlVal(panelHandle, PANEL_Mz, LC_FM_loads[5]);

return;
}
//********************************************************
// DSN_SHIFT                : Will step data along an array
//                                          : For all data that can be plotted
//********************************************************
void DSN_SHIFT(void)
{
        Shift (step,                NUM, 1, step);
        Shift (volt_Fx,     NUM, 1, volt_Fx);
        Shift (volt_Fy,     NUM, 1, volt_Fy);
        Shift (volt_Fz,     NUM, 1, volt_Fz);
        Shift (volt_Mx,     NUM, 1, volt_Mx);
        Shift (volt_My,     NUM, 1, volt_My);
        Shift (volt_Mz,     NUM, 1, volt_Mz);
        Shift (Fx,          NUM, 1, Fx);
        Shift (Fy,          NUM, 1, Fy);
        Shift (Fz,          NUM, 1, Fz);
        Shift (Mx,                  NUM, 1, Mx);
        Shift (My,                  NUM, 1, My);
        Shift (Mz,                  NUM, 1, Mz);
        Shift (position, NUM, 1, position);

return;
}

//********************************************************
// OnOff_Graphs             : For turning graphs ON and OFF
//********************************************************
int CVICALLBACK OnOff_Graphs (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{  int test;
        switch (event)
                {
```

```
        case EVENT_COMMIT:
            switch(control)
                {
                case G_SETUP_OnOff_G_1:
                    GetCtrlVal (g_Handle, G_SETUP_OnOff_G_1, &test);
                    if(test)
                     {
                     SetCtrlAttribute (panelHandle, PANEL_G_1, ATTR_DIMMED, 0);
                     OnOff_G1 = 1;
                     }
                    else
                     {
                     SetCtrlAttribute (panelHandle, PANEL_G_1, ATTR_DIMMED, 1);
                     OnOff_G1 = 0;
                     }
                    break;
                case G_SETUP_OnOff_G_2:
                    GetCtrlVal (g_Handle, G_SETUP_OnOff_G_2, &test);
                    if(test)
                     {
                     SetCtrlAttribute (panelHandle, PANEL_G_2, ATTR_DIMMED, 0);
                     OnOff_G2 = 1;
                     }
                    else
                     {
                     SetCtrlAttribute (panelHandle, PANEL_G_2, ATTR_DIMMED, 1);
                     OnOff_G2 = 0;
                     }
                    break;
                case G_SETUP_OnOff_G_3:
                    GetCtrlVal (g_Handle, G_SETUP_OnOff_G_3, &test);
                    if(test)
                     {
                     SetCtrlAttribute (panelHandle, PANEL_G_3, ATTR_DIMMED, 0);
                     OnOff_G3 = 1;
                     }
                    else
                     {
                     SetCtrlAttribute (panelHandle, PANEL_G_3, ATTR_DIMMED, 1);
                     OnOff_G3 = 0;
                     }
                    break;
                case G_SETUP_OnOff_G_4:
                    GetCtrlVal (g_Handle, G_SETUP_OnOff_G_4, &test);
                    if(test)
                     {
                     SetCtrlAttribute (panelHandle, PANEL_G_4, ATTR_DIMMED, 0);
                     OnOff_G4 = 1;
                     }
                    else
                     {
                     SetCtrlAttribute (panelHandle, PANEL_G_4, ATTR_DIMMED, 1);
                     OnOff_G4 = 0;
                     }
                    break;
                }
            break;
        case EVENT_RIGHT_CLICK:

            break;
        }
    return 0;
}
//******************************************************
// DSN_Graph                    : Do the graphing
//******************************************************
void DSN_Graph(void)
{
// GPARH #1
if(OnOff_G1)
{
DSN_Graph_Select(plotVar_G_1, PANEL_G_1);
    if(plotVar_G_1>=0)
    {
    SetAxisScalingMode (panelHandle, PANEL_G_1, VAL_XAXIS, VAL_MANUAL,
                                                    step[0]-X_Range_G_1, step[0]);
    if(Y_Mode_G_1)
     SetAxisScalingMode (panelHandle, PANEL_G_1, VAL_LEFT_YAXIS,
                                        VAL_AUTOSCALE,0 ,0 );
        else
         SetAxisScalingMode (panelHandle, PANEL_G_1, VAL_LEFT_YAXIS,
                                                  VAL_MANUAL, Y_Min_G_1, Y_Max_G_1);
        }
}
// GPARH #2
if(OnOff_G2)
{
DSN_Graph_Select(plotVar_G_2, PANEL_G_2);
        if(plotVar_G_2>=0)
        {
        SetAxisScalingMode (panelHandle, PANEL_G_2, VAL_XAXIS, VAL_MANUAL,
                                                                  step[0]-X_Range_G_2, step[0]);
        if(Y_Mode_G_2)
         SetAxisScalingMode (panelHandle, PANEL_G_2, VAL_LEFT_YAXIS,
                                                  VAL_AUTOSCALE,0 ,0 );
        else
         SetAxisScalingMode (panelHandle, PANEL_G_2, VAL_LEFT_YAXIS,
                                                  VAL_MANUAL, Y_Min_G_2, Y_Max_G_2);
        }
}
// GPARH #3
if(OnOff_G3)
{
DSN_Graph_Select(plotVar_G_3, PANEL_G_3);
        if(plotVar_G_3>=0)
        {
        SetAxisScalingMode (panelHandle, PANEL_G_3, VAL_XAXIS, VAL_MANUAL,
                                                                  step[0]-X_Range_G_3, step[0]);
        if(Y_Mode_G_3)
         SetAxisScalingMode (panelHandle, PANEL_G_3, VAL_LEFT_YAXIS,
                                                  VAL_AUTOSCALE,0 ,0 );
        else
         SetAxisScalingMode (panelHandle, PANEL_G_3, VAL_LEFT_YAXIS,
                                                  VAL_MANUAL, Y_Min_G_3, Y_Max_G_3);
        }
}
// GPARH #4
if(OnOff_G4)
{
DSN_Graph_Select(plotVar_G_4, PANEL_G_4);

        if(plotVar_G_4>=0)
        {
        SetAxisScalingMode (panelHandle, PANEL_G_4, VAL_XAXIS, VAL_MANUAL,
                                                                  step[0]-X_Range_G_4, step[0]);
        if(Y_Mode_G_4)
         SetAxisScalingMode (panelHandle, PANEL_G_4, VAL_LEFT_YAXIS,
                                                  VAL_AUTOSCALE,0 ,0 );
        else
         SetAxisScalingMode (panelHandle, PANEL_G_4, VAL_LEFT_YAXIS,
                                                  VAL_MANUAL, Y_Min_G_4, Y_Max_G_4);
        }
}
return;
}
//*******************************************************
// DSN_Graph_Select                    : Select Grap to be plotted
//              : NEED TO EDIT THIS
//*******************************************************
void DSN_Graph_Select(int plotVar, int Panel_Graph)
{
switch(plotVar)
        {       case -1:
                        DeleteGraphPlot (panelHandle, Panel_Graph, -1,VAL_IMMEDIATE_DRAW);
                        break;
                case 0:
                        DeleteGraphPlot (panelHandle, Panel_Graph, -1,VAL_IMMEDIATE_DRAW);
                        PlotXY (panelHandle, Panel_Graph, step, Fx, NUM, VAL_DOUBLE,
                                        VAL_DOUBLE, VAL_FAT_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
                                        VAL_RED);
                        break;
                case 1:
                        DeleteGraphPlot (panelHandle, Panel_Graph, -1,VAL_IMMEDIATE_DRAW);
                        PlotXY (panelHandle, Panel_Graph, step, Fy, NUM, VAL_DOUBLE,
                                        VAL_DOUBLE, VAL_FAT_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
                                        VAL_RED);
                        break;
                case 2:
                        DeleteGraphPlot (panelHandle, Panel_Graph, -1,VAL_IMMEDIATE_DRAW);
                        PlotXY (panelHandle, Panel_Graph, step, Fz, NUM, VAL_DOUBLE,
                                        VAL_DOUBLE, VAL_FAT_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
                                        VAL_RED);
                        break;
                case 3:
                        DeleteGraphPlot (panelHandle, Panel_Graph, -1,VAL_IMMEDIATE_DRAW);
                        PlotXY (panelHandle, Panel_Graph, step, Mx, NUM, VAL_DOUBLE,
                                        VAL_DOUBLE, VAL_FAT_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
                                        VAL_RED);
                        break;
                case 4:
                        DeleteGraphPlot (panelHandle, Panel_Graph, -1,VAL_IMMEDIATE_DRAW);
                        PlotXY (panelHandle, Panel_Graph, step, My, NUM, VAL_DOUBLE,
                                        VAL_DOUBLE, VAL_FAT_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
                                        VAL_RED);
                case 5:
                        DeleteGraphPlot (panelHandle, Panel_Graph, -1,VAL_IMMEDIATE_DRAW);
                        PlotXY (panelHandle, Panel_Graph, step, Mz, NUM, VAL_DOUBLE,
                                        VAL_DOUBLE, VAL_FAT_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
                                        VAL_RED);
                case 6:
                        DeleteGraphPlot (panelHandle, Panel_Graph, -1,VAL_IMMEDIATE_DRAW);
                        PlotXY (panelHandle, Panel_Graph, step, position, NUM, VAL_DOUBLE,
                                        VAL_DOUBLE, VAL_FAT_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
                                        VAL_RED);
                        break;

        }
return;
}
//*******************************************************
// GRAPHS          : Call to setup graps
//*******************************************************
int CVICALLBACK DSN_SETUP_G (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        DSN_GraphSetup();
                        break;
                case EVENT_RIGHT_CLICK:

                        break;
                }
        return 0;
}
//*******************************************************
// DSN_GraphName                        : Setup graphs
//              : NEED TO EDIT THIS
//*******************************************************
void DSN_GraphName(int Val, int GRAPH, int L1, int T1, int L2, int T2,
                                                                        int L3, int T3, int L4, int T4)
{
        switch(Val)
                {
                case -1:
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_ACTIVE_YAXIS, VAL_LEFT_YAXIS);
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_YNAME, "Not Plotting");
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_ACTIVE_YAXIS, VAL_RIGHT_YAXIS);
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_YNAME, "");
                        SetCtrlAttribute (g_Handle, L1, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                 SetCtrlVal (g_Handle, T1, "");
                        SetCtrlAttribute (g_Handle, L2, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                 SetCtrlVal (g_Handle, T2, "");
                        SetCtrlAttribute (g_Handle, L3, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                 SetCtrlVal (g_Handle, T3, "");
                        SetCtrlAttribute (g_Handle, L4, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                 SetCtrlVal (g_Handle, T4, "");
                        break;
                case 0:
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_ACTIVE_YAXIS, VAL_LEFT_YAXIS);
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_YNAME, "Fx");
                        SetCtrlAttribute (g_Handle, L1, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                 SetCtrlVal (g_Handle, T1, "");
                        SetCtrlAttribute (g_Handle, L2, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                 SetCtrlVal (g_Handle, T2, "");
                        SetCtrlAttribute (g_Handle, L3, ATTR_FRAME_COLOR, VAL_LT_GRAY);
```

```
                                SetCtrlVal (g_Handle, T3, "");
                        SetCtrlAttribute (g_Handle, L4, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T4, "");
                        break;
                case 1:
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_ACTIVE_YAXIS, VAL_LEFT_YAXIS);
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_YNAME, "Fy");
                        SetCtrlAttribute (g_Handle, L1, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T1, "");
                        SetCtrlAttribute (g_Handle, L2, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T2, "");
                        SetCtrlAttribute (g_Handle, L3, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T3, "");
                        SetCtrlAttribute (g_Handle, L4, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T4, "");
                        break;
                case 2:
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_ACTIVE_YAXIS, VAL_LEFT_YAXIS);
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_YNAME, "Fz");
                        SetCtrlAttribute (g_Handle, L1, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T1, "");
                        SetCtrlAttribute (g_Handle, L2, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T2, "");
                        SetCtrlAttribute (g_Handle, L3, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T3, "");
                        SetCtrlAttribute (g_Handle, L4, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T4, "");
                        break;
                case 3:
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_ACTIVE_YAXIS, VAL_LEFT_YAXIS);
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_YNAME, "Mx");
                        SetCtrlAttribute (g_Handle, L1, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T1, "");
                        SetCtrlAttribute (g_Handle, L2, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T2, "");
                        SetCtrlAttribute (g_Handle, L3, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T3, "");
                        SetCtrlAttribute (g_Handle, L4, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T4, "");
                        break;
                case 4:
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_ACTIVE_YAXIS, VAL_LEFT_YAXIS);
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_YNAME, "My");
                        SetCtrlAttribute (g_Handle, L1, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T1, "");
                        SetCtrlAttribute (g_Handle, L2, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T2, "");
                        SetCtrlAttribute (g_Handle, L3, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T3, "");
                        SetCtrlAttribute (g_Handle, L4, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T4, "");
                        break;
                case 5:
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_ACTIVE_YAXIS, VAL_LEFT_YAXIS);
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_YNAME, "Mz");
                        SetCtrlAttribute (g_Handle, L1, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T1, "");
                        SetCtrlAttribute (g_Handle, L2, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T2, "");
                        SetCtrlAttribute (g_Handle, L3, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T3, "");
                        SetCtrlAttribute (g_Handle, L4, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T4, "");
                        break;
                case 6:
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_ACTIVE_YAXIS, VAL_LEFT_YAXIS);
                        SetCtrlAttribute (panelHandle, GRAPH, ATTR_YNAME, "position");
                        SetCtrlAttribute (g_Handle, L1, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T1, "");
                        SetCtrlAttribute (g_Handle, L2, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T2, "");
                        SetCtrlAttribute (g_Handle, L3, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T3, "");
                        SetCtrlAttribute (g_Handle, L4, ATTR_FRAME_COLOR, VAL_LT_GRAY);
                                SetCtrlVal (g_Handle, T4, "");
                        break;
                }
return;
}
```

```
//********************************************************
// DSN_GraphSetup                          : Setup graphs
//********************************************************
void DSN_GraphSetup(void)
{
// Update the variables from the panels
DSN_Save_Vars();
//Read Graph #1
            if(Y_Mode_G_1)
              {
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Mode_G_1,     ATTR_LABEL_TEXT, "Auto");
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Min_G_1,       ATTR_DIMMED, 1);
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Max_G_1,       ATTR_DIMMED, 1);
              }
            else
                        {
               SetCtrlAttribute (g_Handle, G_SETUP_Y_Mode_G_1, ATTR_LABEL_TEXT, "Fixed");
                        SetCtrlAttribute (g_Handle, G_SETUP_Y_Min_G_1,          ATTR_DIMMED, 0);
               SetCtrlAttribute (g_Handle, G_SETUP_Y_Max_G_1,       ATTR_DIMMED, 0);
               }
                        DSN_GraphName(plotVar_G_1, PANEL_G_1,G_SETUP_G1_L_1,G_SETUP_G1_T_1,

G_SETUP_G1_L_2,G_SETUP_G1_T_2,

G_SETUP_G1_L_3,G_SETUP_G1_T_3,

G_SETUP_G1_L_4,G_SETUP_G1_T_4);
//Read Graph #2
            if(Y_Mode_G_2)
              {
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Mode_G_2,     ATTR_LABEL_TEXT, "Auto");
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Min_G_2,       ATTR_DIMMED, 1);
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Max_G_2,       ATTR_DIMMED, 1);
              }
            else
                        {
               SetCtrlAttribute (g_Handle, G_SETUP_Y_Mode_G_2, ATTR_LABEL_TEXT, "Fixed");
                        SetCtrlAttribute (g_Handle, G_SETUP_Y_Min_G_2,          ATTR_DIMMED, 0);
               SetCtrlAttribute (g_Handle, G_SETUP_Y_Max_G_2,       ATTR_DIMMED, 0);
               }
               DSN_GraphName(plotVar_G_2, PANEL_G_2,G_SETUP_G2_L_1,G_SETUP_G2_T_1,

G_SETUP_G2_L_2,G_SETUP_G2_T_2,

G_SETUP_G2_L_3,G_SETUP_G2_T_3,

G_SETUP_G2_L_4,G_SETUP_G2_T_4);
//Read Graph #3
            if(Y_Mode_G_3)
              {
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Mode_G_3,     ATTR_LABEL_TEXT, "Auto");
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Min_G_3,       ATTR_DIMMED, 1);
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Max_G_3,       ATTR_DIMMED, 1);
              }
            else
                        {
               SetCtrlAttribute (g_Handle, G_SETUP_Y_Mode_G_3, ATTR_LABEL_TEXT, "Fixed");
                        SetCtrlAttribute (g_Handle, G_SETUP_Y_Min_G_3,          ATTR_DIMMED, 0);
               SetCtrlAttribute (g_Handle, G_SETUP_Y_Max_G_3,       ATTR_DIMMED, 0);
               }
               DSN_GraphName(plotVar_G_3, PANEL_G_3,G_SETUP_G3_L_1,G_SETUP_G3_T_1,

G_SETUP_G3_L_2,G_SETUP_G3_T_2,

G_SETUP_G3_L_3,G_SETUP_G3_T_3,

G_SETUP_G3_L_4,G_SETUP_G3_T_4);
//Read Graph #4
            if(Y_Mode_G_4)
              {
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Mode_G_4,     ATTR_LABEL_TEXT, "Auto");
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Min_G_4,       ATTR_DIMMED, 1);
              SetCtrlAttribute (g_Handle, G_SETUP_Y_Max_G_4,       ATTR_DIMMED, 1);
              }
            else
                        {
               SetCtrlAttribute (g_Handle, G_SETUP_Y_Mode_G_4, ATTR_LABEL_TEXT, "Fixed");
                        SetCtrlAttribute (g_Handle, G_SETUP_Y_Min_G_4,          ATTR_DIMMED, 0);
               SetCtrlAttribute (g_Handle, G_SETUP_Y_Max_G_4,       ATTR_DIMMED, 0);
```

```
          }
          DSN_GraphName(plotVar_G_4, PANEL_G_4,G_SETUP_G4_L_1,G_SETUP_G4_T_1,

G_SETUP_G4_L_2,G_SETUP_G4_T_2,

G_SETUP_G4_L_3,G_SETUP_G4_T_3,

G_SETUP_G4_L_4,G_SETUP_G4_T_4);
return;
}

//*********************************************************
//                      COM #1 - Motor Drive Control
//*********************************************************
// M_COM_OPEN                      : Open the  motor com port
//*********************************************************

int CVICALLBACK M_COM_OPEN (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        DSN_M_COM_OPEN();
                        break;
                case EVENT_RIGHT_CLICK:
                        break;
                }
        return 0;
}


//*********************************************************
// DSN_M_COM_OPEN                          : Open the Motor com port
//*********************************************************
void DSN_M_COM_OPEN(void)
{
        int test;
        SetCtrlVal (com_Handle, COM_SP_STRING_1, '\0');
        SetCtrlVal (com_Handle, COM_SP_STRING_2, '\0');
        SetCtrlVal (com_Handle, COM_SP_STRING_3, '\0');
        SetCtrlVal (com_Handle, COM_SP_STRING_4, '\0');  ///***//
        rmd2[0]='\0';
        DSN_GetMConfig();

        M_port_open = 0;  /* initialize flag to 0 - unopened */
  DisableBreakOnLibraryErrors ();
  RS232Error = OpenComConfig (M_comport, "", M_baudrate, M_parity,
                M_databits, M_stopbits, M_inputq, M_outputq);
  SetCtrlVal          (com_Handle, COM_NUM,   RS232Error);
  EnableBreakOnLibraryErrors ();

  if (RS232Error)
        {
        DisplayRS232Error ();
        }
  else
    {
    M_port_open = 1;
    SetCtrlVal        (com_Handle, COM_NUM,   M_port_open);
    GetCtrlVal (com_Handle, COM_M_XMODE, &M_xmode);
    SetXMode (M_comport, M_xmode);
    GetCtrlVal (com_Handle, COM_M_CTS, &M_ctsmode);
    SetCTSMode (M_comport, M_ctsmode);
    GetCtrlVal (com_Handle, COM_M_TIMEOUT, &M_timeout);
    SetComTime (M_comport, M_timeout);
    }

        //SetCtrlVal (com_Handle, COM_M_STRING_2, rmd2);

        if (M_port_open)
        {
        SetCtrlAttribute (com_Handle,  COM_M_LED,   ATTR_DIMMED, 0);
    SetCtrlAttribute (panelHandle, PANEL_M_LED, ATTR_DIMMED, 0);


    test=1;
    if(test)
        {
```

```
        SetCtrlVal (com_Handle,  COM_M_LED,   1);
        SetCtrlVal (panelHandle, PANEL_M_LED, 1);
                }
        else
                {
                SetCtrlVal (com_Handle,  COM_M_LED,   0);
        SetCtrlVal (panelHandle, PANEL_M_LED, 0);
                }

        }

return;
}

//*********************************************************
// DSN_GetMConfig              : Load vars with M config from screen
//*********************************************************
void DSN_GetMConfig (void)
{

        DSN_Save_Vars();

  #ifdef _NI_unix_
    M_devicename[0]=0;
  #else
    GetLabelFromIndex (com_Handle, COM_M_COM, M_portindex,
           M_devicename);
  #endif
}

//*********************************************************
//                             COM #2 - Load Cell
//*********************************************************
// LC_OPEN                             : Open Load Cell the com port
//*********************************************************
int CVICALLBACK LC_COM_OPEN (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        DSN_LC_OPEN();
                        break;
                case EVENT_RIGHT_CLICK:
                        break;
                }
        return 0;
}
//*********************************************************
// DSN_LC_OPEN                         : Open the com port to the Load Cell
//*********************************************************
void DSN_LC_OPEN(void)
{  int test;
        SetCtrlVal (com_Handle, COM_SP_STRING_1, '\0');
        SetCtrlVal (com_Handle, COM_SP_STRING_2, '\0');
        SetCtrlVal (com_Handle, COM_SP_STRING_3, '\0');
        SetCtrlVal (com_Handle, COM_SP_STRING_4, '\0');
        rmd2[0]='\0';
        DSN_GetLCConfig();

        LC_port_open = 0;  /* initialize flag to 0 - unopened */
  DisableBreakOnLibraryErrors ();
  RS232Error = OpenComConfig (LC_comport, "", LC_baudrate, LC_parity,
                LC_databits, LC_stopbits, LC_inputq, LC_outputq);
  SetCtrlVal          (com_Handle, COM_NUM,    RS232Error);
  EnableBreakOnLibraryErrors ();

  if (RS232Error)
        {
        DisplayRS232Error ();
        }
  else
    {
    LC_port_open = 1;
    SetCtrlVal       (com_Handle, COM_NUM,    LC_port_open);
    GetCtrlVal (com_Handle, COM_LC_XMODE, &LC_xmode);
    SetXMode (LC_comport, LC_xmode);
    GetCtrlVal (com_Handle, COM_LC_CTS, &LC_ctsmode);
    SetCTSMode (LC_comport, LC_ctsmode);
    GetCtrlVal (com_Handle, COM_LC_TIMEOUT, &LC_timeout);
    SetComTime (LC_comport, LC_timeout);
    }

        //SetCtrlVal (com_Handle, COM_LC_STRING_2, rmd2);

        if (LC_port_open)
        {
        SetCtrlAttribute (com_Handle,  COM_LC_LED,   ATTR_DIMMED, 0);
                SetCtrlAttribute (panelHandle, PANEL_LC_LED, ATTR_DIMMED, 0);


    test=1;
    if(test)
        {
        SetCtrlVal (com_Handle,  COM_LC_LED,   1);
        SetCtrlVal (panelHandle, PANEL_LC_LED, 1);
                }
        else
                {
                SetCtrlVal (com_Handle,  COM_LC_LED,   0);
        SetCtrlVal (panelHandle, PANEL_LC_LED, 0);
                }

        }

return;
}

//******************************************************
// DSN_GetLCConfig          : Load vars with SP config from screen
//******************************************************
void DSN_GetLCConfig (void)
{

        DSN_Save_Vars();


  #ifdef _NI_unix_
    LC_devicename[0]=0;
  #else
    GetLabelFromIndex (com_Handle, COM_LC_COM, LC_portindex,
            LC_devicename);
  #endif
}


//******************************************************
//                        COM #3 - Camera Program Control
//******************************************************
// CAM_COM_OPEN                          : Open the  Camera Control com port
//******************************************************

int CVICALLBACK CAM_COM_OPEN (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        DSN_CAM_OPEN();
                        break;
                case EVENT_RIGHT_CLICK:
                        break;
                }
        return 0;
}


//******************************************************
// DSN_CAM_COM_OPEN                      : Open the Camera control com port
//******************************************************
void DSN_CAM_OPEN(void)
{
        int test;
        SetCtrlVal (com_Handle, COM_SP_STRING_1, '\0');
        SetCtrlVal (com_Handle, COM_SP_STRING_2, '\0');
        SetCtrlVal (com_Handle, COM_SP_STRING_3, '\0');
        SetCtrlVal (com_Handle, COM_SP_STRING_4, '\0');  ///***//
        rmd2[0]='\0';
        DSN_GetMConfig();

        M_port_open = 0;  /* initialize flag to 0 - unopened */
  DisableBreakOnLibraryErrors ();
  RS232Error = OpenComConfig (CAM_comport, "", CAM_baudrate, CAM_parity,
                CAM_databits, CAM_stopbits, CAM_inputq, CAM_outputq);
  SetCtrlVal        (com_Handle, COM_NUM,   RS232Error);
  EnableBreakOnLibraryErrors ();

  if (RS232Error)
        {
        DisplayRS232Error ();
        }
  else
    {
    CAM_port_open = 1;
    SetCtrlVal      (com_Handle, COM_NUM,   CAM_port_open);
    GetCtrlVal (com_Handle, COM_CAM_XMODE, &CAM_xmode);
    SetXMode (CAM_comport, CAM_xmode);
    GetCtrlVal (com_Handle, COM_CAM_CTS, &CAM_ctsmode);
    SetCTSMode (CAM_comport, CAM_ctsmode);
    GetCtrlVal (com_Handle, COM_CAM_TIMEOUT, &CAM_timeout);
    SetComTime (CAM_comport, CAM_timeout);
    }

        //SetCtrlVal (com_Handle, COM_CAM_STRING_2, rmd2);

        if (CAM_port_open)
        {
        SetCtrlAttribute (com_Handle,  COM_CAM_LED,   ATTR_DIMMED, 0);
    SetCtrlAttribute (panelHandle, PANEL_CAM_LED, ATTR_DIMMED, 0);


    test=1;
    if(test)
        {
        SetCtrlVal (com_Handle,  COM_CAM_LED,   1);
        SetCtrlVal (panelHandle, PANEL_CAM_LED, 1);
                }
        else
                {
                SetCtrlVal (com_Handle,  COM_CAM_LED,   0);
        SetCtrlVal (panelHandle, PANEL_CAM_LED, 0);
                }

        }

return;
}

//*******************************************************
// DSN_GetCAMConfig          : Load vars with CAM config from screen
//*******************************************************
void DSN_GetCAMConfig (void)
{

        DSN_Save_Vars();

  #ifdef _NI_unix_
    CAM_devicename[0]=0;
  #else
    GetLabelFromIndex (com_Handle, COM_CAM_COM, CAM_portindex,
           CAM_devicename);
  #endif
}


//*******************************************************
// DisplayRS232Error            : Display error information to the user.
//*******************************************************
void DisplayRS232Error (void)
{
  char ErrorMessage[200];
  switch (RS232Error)
    {
    default :
      if (RS232Error < 0)
        {
        Fmt (ErrorMessage, "%s<RS232 error number %i", RS232Error);
```

```
            MessagePopup ("RS232 Message", ErrorMessage);
            }
        break;
    case 0  :
        MessagePopup ("RS232 Message", "No errors.");
        break;
    case -2 :
        Fmt (ErrorMessage, "%s", "Connection failed. \n"
            "Check port settings.");
        MessagePopup ("RS232 Message", ErrorMessage);
        break;
    case -3 :
        Fmt (ErrorMessage, "%s", "No port is open.\n"
            "Check port settings.");
        MessagePopup ("RS232 Message", ErrorMessage);
        break;
    case -99 :
        Fmt (ErrorMessage, "%s", "Timeout error.\n"
            "Check port settings.");
        MessagePopup ("RS232 Message", ErrorMessage);
        break;
    }
}



//*********************************************************
// Load_Config      : Load variables from config file
//                                                  : Update screen from variables
//*********************************************************
void CVICALLBACK Load_Config (int menuBar, int menuItem, void *callbackData,
                int panel)
{
        DSN_Load_Config();
        DSN_Load_Vars();
}
//*********************************************************
// Save_Config      : Save screen to variables
//                                                  : Save variables to config file
//*********************************************************
void CVICALLBACK Save_Config (int menuBar, int menuItem, void *callbackData,
                int panel)
{
        DSN_Save_Vars();
        DSN_Save_Config(0);
}
//*********************************************************
// Save_Config_As   : Save screen to variables
//                                                  : Save variables to config file
//                                                  : Allow the config file to be selected
//*********************************************************
void CVICALLBACK Save_Config_As (int menuBar, int menuItem, void *callbackData,
                int panel)
{
        DSN_Save_Vars();
        DSN_Save_Config(1);
}
//*********************************************************
// DSN_Save_Vars    : Save screen to variables
//*********************************************************
void DSN_Save_Vars(void)
{  int i;
        double J_temp,ALPHA_temp;

        GetCtrlVal(panelHandle, PANEL_LOG_COM1,  &Is_COM1);
        GetCtrlVal(panelHandle, PANEL_LOG_COM2,  &Is_COM2);
        GetCtrlVal(panelHandle, PANEL_CAM_COM,              &Is_COM3);

        GetCtrlVal(panelHandle, PANEL_LOG_Graph,  &Is_Graph);
        GetCtrlVal(panelHandle, PANEL_LOG_LOGtoFile,&Is_Log);
// GRAPH PANEL
        //Read Graph #1
        GetCtrlVal( g_Handle,       G_SETUP_Var_G_1,                &plotVar_G_1);
        GetCtrlVal( g_Handle,       G_SETUP_X_Range_G_1,    &X_Range_G_1);
        GetCtrlVal( g_Handle,       G_SETUP_Y_Mode_G_1,     &Y_Mode_G_1);
        GetCtrlVal( g_Handle,       G_SETUP_Y_Min_G_1,              &Y_Min_G_1);
        GetCtrlVal( g_Handle,       G_SETUP_Y_Max_G_1,              &Y_Max_G_1);
        if (Y_Min_G_1 >= Y_Max_G_1)
```

```
                        Y_Min_G_1 = Y_Max_G_1-0.5;
                        SetCtrlVal( g_Handle,        G_SETUP_Y_Min_G_1,                      Y_Min_G_1);
            //Read Graph #2
            GetCtrlVal( g_Handle,        G_SETUP_Var_G_2,                    &plotVar_G_2);
            GetCtrlVal( g_Handle,        G_SETUP_X_Range_G_2,    &X_Range_G_2);
            GetCtrlVal( g_Handle,        G_SETUP_Y_Mode_G_2,     &Y_Mode_G_2);
            GetCtrlVal( g_Handle,        G_SETUP_Y_Min_G_2,                  &Y_Min_G_2);
            GetCtrlVal( g_Handle,        G_SETUP_Y_Max_G_2,                  &Y_Max_G_2);
            if (Y_Min_G_2 >= Y_Max_G_2)
                        Y_Min_G_2 = Y_Max_G_2-0.5;
                        SetCtrlVal( g_Handle,        G_SETUP_Y_Min_G_2,                      Y_Min_G_2);
            //Read Graph #3
            GetCtrlVal( g_Handle,        G_SETUP_Var_G_3,                    &plotVar_G_3);
            GetCtrlVal( g_Handle,        G_SETUP_X_Range_G_3,    &X_Range_G_3);
            GetCtrlVal( g_Handle,        G_SETUP_Y_Mode_G_3,     &Y_Mode_G_3);
            GetCtrlVal( g_Handle,        G_SETUP_Y_Min_G_3,                  &Y_Min_G_3);
            GetCtrlVal( g_Handle,        G_SETUP_Y_Max_G_3,                  &Y_Max_G_3);
            if (Y_Min_G_3 >= Y_Max_G_3)
                        Y_Min_G_3 = Y_Max_G_3-0.5;
                        SetCtrlVal( g_Handle,        G_SETUP_Y_Min_G_3,                      Y_Min_G_3);
            //Read Graph #4
            GetCtrlVal( g_Handle,        G_SETUP_Var_G_4,                    &plotVar_G_4);
            GetCtrlVal( g_Handle,        G_SETUP_X_Range_G_4,    &X_Range_G_4);
            GetCtrlVal( g_Handle,        G_SETUP_Y_Mode_G_4,     &Y_Mode_G_4);
            GetCtrlVal( g_Handle,        G_SETUP_Y_Min_G_4,                  &Y_Min_G_4);
            GetCtrlVal( g_Handle,        G_SETUP_Y_Max_G_4,                  &Y_Max_G_4);
            if (Y_Min_G_4 >= Y_Max_G_4)
                        Y_Min_G_4 = Y_Max_G_4-0.5;
                        SetCtrlVal( g_Handle,        G_SETUP_Y_Min_G_4,                      Y_Min_G_4);

// COM PANEL
            GetCtrlVal( com_Handle, COM_M_COM,                                   &M_comport);
   GetCtrlVal( com_Handle, COM_M_BR,                                 &M_baudrate);
   GetCtrlVal( com_Handle, COM_M_P,                                  &M_parity);
   GetCtrlVal( com_Handle, COM_M_DB,                                 &M_databits);
   GetCtrlVal( com_Handle, COM_M_SB,                                 &M_stopbits);
   GetCtrlVal( com_Handle, COM_M_INPUTQ,                 &M_inputq);
   GetCtrlVal( com_Handle, COM_M_OUTPUTQ,                            &M_outputq);
   GetCtrlIndex(com_Handle, COM_M_COM,                   &M_portindex);

   GetCtrlVal( com_Handle, COM_LC_COM,                               &LC_comport);
   GetCtrlVal( com_Handle, COM_LC_BR,                                            &LC_baudrate);
   GetCtrlVal( com_Handle, COM_LC_P,                                             &LC_parity);
   GetCtrlVal( com_Handle, COM_LC_DB,                                            &LC_databits);
   GetCtrlVal( com_Handle, COM_LC_SB,                                            &LC_stopbits);
   GetCtrlVal( com_Handle, COM_LC_INPUTQ,                                        &LC_inputq);
   GetCtrlVal( com_Handle, COM_LC_OUTPUTQ,                                       &LC_outputq);
   GetCtrlIndex(com_Handle, COM_LC_COM,                              &LC_portindex);

            GetCtrlVal( com_Handle, COM_CAM_COM,                                 &CAM_comport);
   GetCtrlVal( com_Handle, COM_CAM_BR,                               &CAM_baudrate);
   GetCtrlVal( com_Handle, COM_CAM_P,                                            &CAM_parity);
   GetCtrlVal( com_Handle, COM_CAM_DB,                               &CAM_databits);
   GetCtrlVal( com_Handle, COM_CAM_SB,                               &CAM_stopbits);
   GetCtrlVal( com_Handle, COM_CAM_INPUTQ,                           &CAM_inputq);
   GetCtrlVal( com_Handle, COM_CAM_OUTPUTQ,                          &CAM_outputq);
   GetCtrlIndex(com_Handle, COM_CAM_COM,                                         &CAM_portindex);

// Load Cell nverse Sensitivty Matrix


            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,1), &ISM_11);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,1), &ISM_12);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,1), &ISM_13);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,1), &ISM_14);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,1), &ISM_15);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,1), &ISM_16);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,2), &ISM_21);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,2), &ISM_22);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,2), &ISM_23);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,2), &ISM_24);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,2), &ISM_25);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,2), &ISM_26);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,3), &ISM_31);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,3), &ISM_32);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,3), &ISM_33);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,3), &ISM_34);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,3), &ISM_35);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,3), &ISM_36);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,4), &ISM_41);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,4), &ISM_42);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,4), &ISM_43);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,4), &ISM_44);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,4), &ISM_45);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,4), &ISM_46);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,5), &ISM_51);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,5), &ISM_52);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,5), &ISM_53);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,5), &ISM_54);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,5), &ISM_55);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,5), &ISM_56);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,6), &ISM_61);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,6), &ISM_62);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,6), &ISM_63);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,6), &ISM_64);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,6), &ISM_65);
            GetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,6), &ISM_66);

//Load Cell Gains

            GetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(1,1), &gain_Fx);
            GetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(2,1), &gain_Fy);
            GetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(3,1), &gain_Fz);
            GetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(4,1), &gain_Mx);
            GetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(5,1), &gain_My);
            GetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(6,1), &gain_Mz);

return;
}
//*****************************************************
// DSN_Load_Vars    : Load variables to the screen
//*****************************************************
void DSN_Load_Vars(void)
{
// MAIN PANEL

// GRAPH PANEL
            //Read Graph #1
            SetCtrlVal( g_Handle,         G_SETUP_Var_G_1,                plotVar_G_1);
            SetCtrlVal( g_Handle,         G_SETUP_X_Range_G_1,    X_Range_G_1);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Mode_G_1,     Y_Mode_G_1);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Min_G_1,              Y_Min_G_1);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Max_G_1,              Y_Max_G_1);
            //Read Graph #2
            SetCtrlVal( g_Handle,         G_SETUP_Var_G_2,                plotVar_G_2);
            SetCtrlVal( g_Handle,         G_SETUP_X_Range_G_2,    X_Range_G_2);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Mode_G_2,     Y_Mode_G_2);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Min_G_2,              Y_Min_G_2);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Max_G_2,              Y_Max_G_2);
            //Read Graph #3
            SetCtrlVal( g_Handle,         G_SETUP_Var_G_3,                plotVar_G_3);
            SetCtrlVal( g_Handle,         G_SETUP_X_Range_G_3,    X_Range_G_3);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Mode_G_3,     Y_Mode_G_3);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Min_G_3,              Y_Min_G_3);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Max_G_3,              Y_Max_G_3);
            //Read Graph #4
            SetCtrlVal( g_Handle,         G_SETUP_Var_G_4,                plotVar_G_4);
            SetCtrlVal( g_Handle,         G_SETUP_X_Range_G_4,    X_Range_G_4);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Mode_G_4,     Y_Mode_G_4);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Min_G_4,              Y_Min_G_4);
            SetCtrlVal( g_Handle,         G_SETUP_Y_Max_G_4,              Y_Max_G_4);


// COM PANEL
            SetCtrlVal( com_Handle, COM_M_COM,                                    M_comport);
   SetCtrlVal( com_Handle, COM_M_BR,                                  M_baudrate);
   SetCtrlVal( com_Handle, COM_M_P,                                   M_parity);
   SetCtrlVal( com_Handle, COM_M_DB,                                  M_databits);
   SetCtrlVal( com_Handle, COM_M_SB,                                  M_stopbits);
   SetCtrlVal( com_Handle, COM_M_INPUTQ,                      M_inputq);
   SetCtrlVal( com_Handle, COM_M_OUTPUTQ,                             M_outputq);
   //SetCtrlIndex(com_Handle, COM_SP_COM,                     SP_portindex);

            SetCtrlVal( com_Handle, COM_LC_COM,                        LC_comport);
   SetCtrlVal( com_Handle, COM_LC_BR,                                 LC_baudrate);
   SetCtrlVal( com_Handle, COM_LC_P,                                  LC_parity);
   SetCtrlVal( com_Handle, COM_LC_DB,                                 LC_databits);
   SetCtrlVal( com_Handle, COM_LC_SB,                                 LC_stopbits);
   SetCtrlVal( com_Handle, COM_LC_INPUTQ,                          LC_inputq);
   SetCtrlVal( com_Handle, COM_LC_OUTPUTQ,                                 LC_outputq);

          SetCtrlVal( com_Handle, COM_CAM_COM,                             CAM_comport);
   SetCtrlVal( com_Handle, COM_CAM_BR,                             CAM_baudrate);
   SetCtrlVal( com_Handle, COM_CAM_P,                                      CAM_parity);
   SetCtrlVal( com_Handle, COM_CAM_DB,                             CAM_databits);
   SetCtrlVal( com_Handle, COM_CAM_SB,                             CAM_stopbits);
   SetCtrlVal( com_Handle, COM_CAM_INPUTQ,                         CAM_inputq);
   SetCtrlVal( com_Handle, COM_CAM_OUTPUTQ,                        CAM_outputq);

//Set Inverse Sensitivty Matrix On GUI

          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,1), ISM_11);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,1), ISM_12);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,1), ISM_13);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,1), ISM_14);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,1), ISM_15);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,1), ISM_16);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,2), ISM_21);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,2), ISM_22);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,2), ISM_23);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,2), ISM_24);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,2), ISM_25);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,2), ISM_26);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,3), ISM_31);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,3), ISM_32);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,3), ISM_33);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,3), ISM_34);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,3), ISM_35);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,3), ISM_36);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,4), ISM_41);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,4), ISM_42);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,4), ISM_43);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,4), ISM_44);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,4), ISM_45);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,4), ISM_46);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,5), ISM_51);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,5), ISM_52);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,5), ISM_53);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,5), ISM_54);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,5), ISM_55);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,5), ISM_56);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(1,6), ISM_61);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(2,6), ISM_62);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(3,6), ISM_63);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(4,6), ISM_64);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(5,6), ISM_65);
          SetTableCellVal(v_corr_Handle, V_CORR_ISM, MakePoint(6,6), ISM_66);

          //Set Gain values on GUI

          SetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(1,1), gain_Fx);
          SetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(2,1), gain_Fy);
          SetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(3,1), gain_Fz);
          SetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(4,1), gain_Mx);
          SetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(5,1), gain_My);
          SetTableCellVal(v_corr_Handle, V_CORR_GAINS, MakePoint(6,1), gain_Mz);

//Put Inverse Sensitivty Matrix values into array
          LC_ISM[0][0] = ISM_11;
          LC_ISM[0][1] = ISM_12;
          LC_ISM[0][2] = ISM_13;
          LC_ISM[0][3] = ISM_14;
          LC_ISM[0][4] = ISM_15;
          LC_ISM[0][5] = ISM_16;
          LC_ISM[1][0] = ISM_21;
          LC_ISM[1][1] = ISM_22;
          LC_ISM[1][2] = ISM_23;
          LC_ISM[1][3] = ISM_24;
          LC_ISM[1][4] = ISM_25;
          LC_ISM[1][5] = ISM_26;
          LC_ISM[2][0] = ISM_31;
          LC_ISM[2][1] = ISM_32;
          LC_ISM[2][2] = ISM_33;
          LC_ISM[2][3] = ISM_34;
          LC_ISM[2][4] = ISM_35;
          LC_ISM[2][5] = ISM_36;
          LC_ISM[3][0] = ISM_41;
```

```
            LC_ISM[3][1] = ISM_42;
            LC_ISM[3][2] = ISM_43;
            LC_ISM[3][3] = ISM_44;
            LC_ISM[3][4] = ISM_45;
            LC_ISM[3][5] = ISM_46;
            LC_ISM[4][0] = ISM_51;
            LC_ISM[4][1] = ISM_52;
            LC_ISM[4][2] = ISM_53;
            LC_ISM[4][3] = ISM_54;
            LC_ISM[4][4] = ISM_55;
            LC_ISM[4][5] = ISM_56;
            LC_ISM[5][0] = ISM_61;
            LC_ISM[5][1] = ISM_62;
            LC_ISM[5][2] = ISM_63;
            LC_ISM[5][3] = ISM_64;
            LC_ISM[5][4] = ISM_65;
            LC_ISM[5][5] = ISM_66;

//Put Gain values into array
            LC_gains[0] = gain_Fx;
            LC_gains[1] = gain_Fy;
            LC_gains[2] = gain_Fz;
            LC_gains[3] = gain_Mx;
            LC_gains[4] = gain_My;
            LC_gains[5] = gain_Mz;



return;
}
//*******************************************************
// DSN_Init2                    : Init some variable and do some setup
//*******************************************************
void DSN_Init2()
{ // Set the colours up

return;
}
//*******************************************************
//*******************************************************
//*******************************************************

//*******************************************************
// DSN_Parse_SP             : Used to parse the syringe pump return string
//*******************************************************

void DSN_Parse_SP(char IN_str[], char prom[], int *vol_sp)
{
            int i=0;
            int j=0;
            int k=0;
            char str[30];
            double temp;

i = strcspn (IN_str, ":<>E");               // Find the position of the Prompt
//SetCtrlVal(lic_Handle,LICOM_Count,  i);

j = strcspn (IN_str, "0123456789"); // Find the position of the first number
k = strcspn (IN_str, "l");                                  // Find the position of the end of the volume units
if (i>1 && (k-j)<30)
            {
            CopyString (str, 0, IN_str, j, k-j);
            (*vol_sp) = atoi(str);
            // Get the Prompt
            CopyString (prom, 0, IN_str, i-1, 2);
            }

}
//*******************************************************
// QB_Read_LC       :             Reads from load cell
//*******************************************************
void QB_Read_LC()
{
  SetCTSMode (LC_comport, LWRS_HWHANDSHAKE_OFF);                                  //Turn off Hardware handshaking

                                          ComWrt (LC_comport, LC_start, StringLength(LC_start));                //Send the start
command

                                          //Read the FIRST byte
```

```
                                bytes_read = ComRd (LC_comport, LC_first_byte, 1);

                                //************* QB: Should I Put Something Here To Check If It's A
"U"???**********//

                                //Read TWELVE bytes

                                bytes_read = ComRd (LC_comport, LC_read_data, 12);

                                len = StringLength(LC_read_data);

                                for (t=0; t<len; t++)
                                {
                                QB_Print_Bin(LC_read_data[t]);                  //Send the first character to
printbin

                                sprintf( LC_word, LC_onebyte);                  //Put the first character's
binary in word

                                t++;
                                QB_Print_Bin(LC_read_data[t]);                  //Send the second character
to printnin

                                strcat(LC_word,LC_onebyte);                             //Concatenate the
binary of the first and second characters into word

                                QB_Extract_Data();
                                }
}

//*********************************************************
// QB_Print_Bin     : Converts an Ascii byte into binary
//*********************************************************
void QB_Print_Bin(unsigned int n)
{
        char b[] = "00000000";                  //Initializes a byte of zeroes

        for (i=0; i<8; i++, n=n/2)
        {
                if (n%2) b[7-i] = '1';                  //Changes zeroes to ones where applicable
                if (i>8) break;

        }

        sprintf(LC_onebyte,b);
        }

//*********************************************************
// QB_Extract_Data              : Extract Load cell voltages and chanel numbers
//*********************************************************
void QB_Extract_Data()
{
        char LC_data_string[15];
        char LC_channel_string[5];
        unsigned int LC_channel;
        char null_term[] = "\0" ;
        unsigned int LC_val=0;
        int overflow;
        int LC_twos_comp;
        double LC_twos_comp_double;

        //Extract the nibbles with the load data and put them in data_string
                                for (i=12; i<16; i++)
                                        {
                                        LC_data_string[i-12] = LC_word[i];
                                        }
                                 for (i=0; i<8; i++)
                                        {
                                        LC_data_string[i+4] = LC_word[i];
                                        }

                                LC_data_string[i+4]= null_term[0];
        //Add null terminator to end of channel_string

                                LC_val = BinStrToUInt (LC_data_string, &overflow);                      //Get the
integer value of the data

                          //Extract the nibble with the channel number and put it in channel)string
                                for (i=8; i<12; i++)
```

```
                                        {
                                                  LC_channel_string[i-8] = LC_word[i];
                                                  }

                                        LC_channel_string[i-8]= null_term[0];
          //Add null terminator to end of channel_string

                                        LC_channel = BinStrToUInt (LC_channel_string, &overflow);          //Get the channel
number

                                        LC_twos_comp = LC_val-2048;
                              //Subtract 2048 to get the twos compliment value

                                        LC_twos_comp_double = LC_twos_comp;
                    //Convert twos compliment into double

                                        LC_volt = LC_twos_comp_double/100;
          //Divide by 100 to get voltage

                                        //Sort voltages according to channel number
                                        switch(LC_channel)
                                        {
                                                  case 0:
                                                            LC_Raw_volts[0] = LC_volt;
                                                            break;
                                                  case 1:
                                                            LC_Raw_volts[1] = LC_volt;
                                                            break;
                                                  case 2:
                                                            LC_Raw_volts[2]= LC_volt;
                                                            break;
                                                  case 3:
                                                            LC_Raw_volts[3] = LC_volt;
                                                            break;
                                                  case 4:
                                                            LC_Raw_volts[4] = LC_volt;
                                                            break;
                                                  case 5:
                                                            LC_Raw_volts[5] = LC_volt;
                                                            break;
                                        }

                                        ProcessSystemEvents ();
}

//*******************************************************
// QB_Convert_FM : Convert voltages into load readings
//*******************************************************
void QB_Convert_FM()
{
                    //CF = Gain*(Excitation Volotage)*1*10^-6        :            from pg. 14 of load cell manual
                    //Excitation voltage = 10 VDC             :            From load cell manual
                    for (i=0;i<6;i++)
                    {
                              LC_CF[i] =  LC_gains[i]*10*pow(10,-6);
                    }


                    for (i=0;i<6;i++)
                    {
                              LC_FM_volts[i] = LC_Raw_volts[i] - LC_Bias_volts[i];
                    }

                    for (i=0;i<6;i++)
                    {         //for  (j=0;j<6;j++)
                                        LC_FM_loads[i] = (LC_ISM[i][0]*LC_FM_volts[0]/LC_CF[0])+
                                                                       (LC_ISM[i][1]*LC_FM_volts[1]/LC_CF[1])+
                                                                       (LC_ISM[i][2]*LC_FM_volts[2]/LC_CF[2])+
                                                                       (LC_ISM[i][3]*LC_FM_volts[3]/LC_CF[3])+
                                                                       (LC_ISM[i][4]*LC_FM_volts[4]/LC_CF[4])+
                                                                       (LC_ISM[i][5]*LC_FM_volts[5]/LC_CF[5]);
                    }


}



//*******************************************************
```

```
// QB_Get_Bias                    :   Get array of bias voltages
//*****************************************************
void QB_Get_Bias()
{
            char LC_data_string[15];
            char LC_channel_string[5];
            unsigned int LC_channel;
            char null_term[] = "\0" ;
            unsigned int LC_val=0;
            int overflow;
            int LC_twos_comp;
            double LC_twos_comp_double;


            //Extract the nibbles with the load data and put them in data_string
                                    for (i=12; i<16; i++)
                                                {
                                                LC_data_string[i-12] = LC_word[i];
                                                }
                                     for (i=0; i<8; i++)
                                                {
                                                LC_data_string[i+4] = LC_word[i];
                                                }

                                    LC_data_string[i+4]= null_term[0];
            //Add null terminator to end of channel_string

                                    LC_val = BinStrToUInt (LC_data_string, &overflow);                              //Get the
integer value of the data

                              //Extract the nibble with the channel number and put it in channel)string
                                    for (i=8; i<12; i++)
                                    {
                                                LC_channel_string[i-8] = LC_word[i];
                                                }

                                    LC_channel_string[i-8]= null_term[0];
             //Add null terminator to end of channel_string

                                    LC_channel = BinStrToUInt (LC_channel_string, &overflow);           //Get the channel
number

                                    LC_twos_comp = LC_val-2048;
                              //Subtract 2048 to get the twos compliment value

                                    LC_twos_comp_double = LC_twos_comp;
                        //Convert twos compliment into double

                                    LC_volt = LC_twos_comp_double/100;
            //Divide by 100 to get voltage

                                    //Sort voltages according to channel number
                                    switch(LC_channel)
                                    {
                                                case 0:
                                                            LC_Bias_volts[0] = LC_volt;
                                                            break;
                                                case 1:
                                                            LC_Bias_volts[1] = LC_volt;
                                                            break;
                                                case 2:
                                                            LC_Bias_volts[2]= LC_volt;
                                                            break;
                                                case 3:
                                                            LC_Bias_volts[3] = LC_volt;
                                                            break;
                                                case 4:
                                                            LC_Bias_volts[4] = LC_volt;
                                                            break;
                                                case 5:
                                                            LC_Bias_volts[5] = LC_volt;
                                                            break;
                                    }

                                    ProcessSystemEvents ();
}

//*****************************************************
```

```
// Jog Up Callback   added by QB 2012 :
//*******************************************************
int CVICALLBACK M_JOG_UP (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    //-----------------------------------------------
    // QB MOTOR DRIVE COMMANDS WITH CARRIAGE RETURN
    //-----------------------------------------------
    char M_enable[4] = "EN";
    char M_jog_command[6] = "MJOG";
    char M_stop[6] = "STOP";
    char M_disable[5] = "DIS";
    char M_op0[10] = "OPMODE 0";
    //M_downflag=0;

    switch (event)
    {
        case EVENT_LEFT_CLICK:

                SetCtrlAttribute(HN_TABHandle[0], TAB_QUICK_M_Up_Button,
ATTR_CMD_BUTTON_COLOR,VAL_DK_GRAY);

                //Flush Input and Output Queues
                FlushInQ (M_comport);
                FlushOutQ(M_comport);

                //Enable the motor drive
                strcat (M_enable,"\r");
                ComWrt (M_comport, M_enable, strlen(M_enable));

                //Put into Operation Mode 0 : digtal velocity
                strcat(M_op0, "\r");
                ComWrt (M_comport, M_op0, strlen(M_op0));

                GetCtrlVal( HN_TABHandle[0], TAB_QUICK_M_JOG_SPEED, &M_sjog );
    //Get the input jog speed from the GUI

                M_sent_len =  sprintf( M_jog, "J %3.2f" , M_sjog);
    //Assemble command          :       J velocity
                strcat (M_jog,"\r");
        //Add the Carriage Return

                ComWrt (M_comport, M_jog, strlen(M_jog));                      //Send
jog command to controller

                QB_M_Feedback(6);
                //Read the feedback from the motor drive

                ProcessSystemEvents();


                QB_M_POS();



            break;

            case EVENT_LEFT_CLICK_UP:

                M_get_pos =0;

                //Flush Input and Output Queues
                FlushInQ (M_comport);
                FlushOutQ(M_comport);

                strcat (M_stop,"\r");
    //Add the Carriage Return
                ComWrt (M_comport, M_stop, strlen(M_stop));              //Stop the motor
movement

                //strcat (M_disable,"\r");
    //Add the Carriage Return
                //ComWrt (M_comport, M_disable, strlen(M_disable));   //Disable the actuator

                QB_M_Feedback(4);
        //Read the feedback from the motor drive

                SetCtrlAttribute(HN_TABHandle[0], TAB_QUICK_M_Up_Button,
ATTR_CMD_BUTTON_COLOR,VAL_PANEL_GRAY);
```

```
                        break;

        }
        return 0;
}

//*******************************************************
// Jog Down Callback added by QB 2012              :
//*******************************************************
 int CVICALLBACK M_JOG_DOWN (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{   int test;

        //------------------------------------------------
        // QB MOTOR DRIVE COMMANDS WITH CARRIAGE RETURN
        //------------------------------------------------
        char M_enable[4] = "EN";
        char M_jog_command[6] = "MJOG";
        char M_stop[6] = "STOP";
        char M_disable[5] = "DIS";
        char M_op0[10] = "OPMODE 0";
        //M_downflag=1;

                switch (event)
        {
                case EVENT_LEFT_CLICK:

                        SetCtrlAttribute(HN_TABHandle[0], TAB_QUICK_M_Down_Button,
ATTR_CMD_BUTTON_COLOR,VAL_DK_GRAY);

                        //Flush Input and Output Queues
                        FlushInQ (M_comport);
                        FlushOutQ(M_comport);

                        //Enable the motor drive
                        strcat (M_enable,"\r");
                        ComWrt (M_comport, M_enable, strlen(M_enable));

                        //Put into Operation Mode 0 : digtal velocity
                        strcat(M_op0, "\r");
                        ComWrt (M_comport, M_op0, strlen(M_op0));

                        GetCtrlVal( HN_TABHandle[0], TAB_QUICK_M_JOG_SPEED, &M_sjog );          //Get the
input jog speed from the GUI

                        M_sent_len =  sprintf( M_jog, "J -%1.2f" , M_sjog);                   //Assemble
command :       J velocity
                        strcat (M_jog,"\r");
        //Add the Carriage Return

                        ComWrt (M_comport, M_jog, strlen(M_jog));                             //Send jog
command to controller

                        QB_M_Feedback(6);
                    //Read the feedback from the motor drive

                        ProcessSystemEvents();

                        QB_M_POS();

                        //hn_tmp = 1;
                        //while (hn_tmp ==1)
                        //{
                        //      hn_tmpp = HN_position ();
                        //      SetCtrlVal (panelHandle, PANEL_text2,hn_tmpp);
                        //}


                        break;

                case EVENT_LEFT_CLICK_UP:

                        M_get_pos =0;
                        //hn_tmp = 0;

                        //Flush Input and Output Queues
                        FlushInQ (M_comport);
                        FlushOutQ(M_comport);
```

```
            strcat (M_stop,"\r");
    //Add the Carriage Return
            ComWrt (M_comport, M_stop, strlen(M_stop));                //Stop the motor movement

            //strcat (M_disable,"\r");                                                    //Add the
Carriage Return
            //ComWrt (M_comport, M_disable, strlen(M_disable));   //Disable the actuator

            QB_M_Feedback(4);
    //Read the feedback from the motor drive

            SetCtrlAttribute(HN_TABHandle[0], TAB_QUICK_M_Down_Button,
ATTR_CMD_BUTTON_COLOR,VAL_PANEL_GRAY);

            break;

    }
    return 0;
}


//****************************************************
// Emergency Stop Callback   :
//****************************************************
int CVICALLBACK LC_E_STOP (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

        QB_Kill_Motor();

            break;
    }
    return 0;
}

//****************************************************
// Emergency Stop Callback   :
//****************************************************
void QB_Kill_Motor()
{
    //Flush Input and Output Queues
    FlushInQ (M_comport);
    FlushOutQ(M_comport);

    ComWrt (M_comport, "S\r", 2);          //Stop movement
    //ComWrt (M_comport, "K\r", 2);        //Disable actuator
    QB_M_Feedback(4);                                    //Read the feedback from the motor drive

}
//****************************************************
// QB_M_POS        : Update Motor Position
//****************************************************
void QB_M_POS(void)
{
    char M_pos_feedback[5] = "PFB";


    //Flush Input and Output Queues
    FlushInQ (M_comport);
    FlushOutQ(M_comport);



    strcat(M_pos_feedback, "\r");

    M_get_pos =1;




    while (M_get_pos == 1)
    {
        M_read_data[0] = '\0';
        ComWrt (M_comport, M_pos_feedback, strlen(M_pos_feedback));                //Send the position
inquiry
```

```
        //Expecting three things back :             "PFB" "position" "-->"

                                                     //Read the "PFB"
        QB_M_Feedback(1);

                                                            //Get the input queu length
        M_inqlen = GetInQLen(M_comport);
        //Read the position
        M_bytes_read = ComRdTerm(M_comport, M_position, M_inqlen, 10);
        CopyString (M_tbox_read_data, 0, M_position, 0, M_bytes_read);
        SetCtrlVal(panelHandle, PANEL_M_FEEDBACK,M_tbox_read_data);
        //SetCtrlVal(panelHandle, PANEL_text,M_position);
        M_pos = atoi(M_position);
            //Convert the position to an integer
        SetCtrlVal(panelHandle, PANEL_M_POSITION, M_pos);
    //Send the position to the GUI

        QB_M_Feedback(1);
                                            //Read the "-->"


        //QB_M_Limits();                                                        //Check if the traversing
stage is within its physical limits

        Delay(0.1);


        ProcessSystemEvents();
        }
}

//*******************************************************
// Cancel Fault Status of Motor Drive
//*******************************************************
int CVICALLBACK M_CLRFAULT (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

        //Flush Input and Output Queues
        FlushInQ (M_comport);
        FlushOutQ(M_comport);

        ComWrt (M_comport, "CLRFAULT\r", 9);        //Send the clear fault status command

        QB_M_Feedback(2);                                                       //Read the feedback from the
motor drive

            break;
    }
    return 0;
}

//*******************************************************
// QB_Update_Vars:          Add load cell voltages,
//                                              load cell readings
//                                              and position readings to log file
//                                              arrays
//*******************************************************
void QB_Update_Vars(void)
{
    volt_Fx[0] = LC_Raw_volts[0];
    volt_Fy[0] = LC_Raw_volts[1];
    volt_Fz[0] = LC_Raw_volts[2];
    volt_Mx[0] = LC_Raw_volts[3];
    volt_My[0] = LC_Raw_volts[4];
    volt_Mz[0] = LC_Raw_volts[5];
    Fx[0] = LC_FM_loads[0];
    Fy[0] = LC_FM_loads[1];
    Fz[0] = LC_FM_loads[2];
    Mx[0] = LC_FM_loads[3];
    My[0] = LC_FM_loads[4];
    Mz[0] = LC_FM_loads[5];
    position[0] = M_pos;
}
//*******************************************************
```

```
//  QB_M_Feedback               :               Get feedback from the motor
//                                                              and display it on GUI
//*******************************************************
void   QB_M_Feedback(int M_fback_num)          //M_fbac_num = number of commands sent to the motor drive
{
            for (i=0; i<M_fback_num; i++)
            {
            M_inqlen = GetInQLen(M_comport);
            //Get the input queu length
            M_bytes_read = ComRdTerm(M_comport, M_read_data, M_inqlen, 10);                    //Read from the input queu
            CopyString (M_tbox_read_data, 0, M_read_data, 0, M_bytes_read);               //Adds the nullterminator to the string
            SetCtrlVal(panelHandle, PANEL_M_FEEDBACK,M_tbox_read_data);                                    //Display in Motor
Feedback Textbox
            }
}
//*******************************************************
//  QB_Clear_Arrays             :               clear values in arrays
//*******************************************************
void QB_Clear_Arrays(void)
{
            Set1D(Fx,NUM, NULL);
            Set1D(Fy,NUM, NULL);
            Set1D(Fz,NUM, NULL);
            Set1D(Mx,NUM, NULL);
            Set1D(My,NUM, NULL);
            Set1D(Mz,NUM, NULL);
            Set1D(position,NUM, NULL);


}

//*****************************************************
// HN_calibration
//to see the relation between rpm and displacement
//*****************************************************


int CVICALLBACK HN_calibration (int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)
{
            char M_enable[4] = "EN";
            char M_jog_command[6] = "MJOG";
            char M_stop[6] = "STOP";
            char M_disable[5] = "DIS";
            char M_op0[10] = "OPMODE 0";
            char hn_command[30];
            char hn_pfb [10] = "PFB";
            float hn_speed;
            float hn_pos;
            float hn_time;
            int dr;
            char message[1000];

            short int hn_len;
            //char tmp[30];

            //char hntmp[10] = "J 1 5000";



            switch (event)
            {
                        case EVENT_COMMIT:





                                                //Flush Input and Output Queues
                                                FlushInQ (M_comport);
                                                FlushOutQ(M_comport);

                                                //Enable the motor drive
                                                strcat (M_enable,"\r");
                                                ComWrt (M_comport, M_enable, strlen(M_enable));

                                                //Put into Operation Mode 0 : digtal velocity
                                                strcat(M_op0, "\r");
                                                ComWrt (M_comport, M_op0, strlen(M_op0));
```

```
                //GetCtrlVal


                GetCtrlVal  (HN_TABHandle[2], TAB_VEL_clbr_speed, &hn_speed);
                GetCtrlVal  (HN_TABHandle[2], TAB_VEL_clbr_time, &hn_time);
                GetCtrlVal (HN_TABHandle[2], TAB_VEL_dr, &dr);

                if ( (int) dr==0)
                {
                    hn_speed *= -1;
                }


                hn_len = sprintf (hn_command, "J %1.2f %.0f",hn_speed, hn_time*1000);

                //sprintf (tmp,"%s",hn_command)   ;
                //SetCtrlVal (HN_clbr_Handle, PANEL_CLBR_AA, tmp);



                strcat (hn_command,"\r");
                ComWrt (M_comport, hn_command, strlen (hn_command));

                DelayWithEventProcessing (hn_time);


                /*
                strcat (hntmp,"\r")     ;
                ComWrt (M_comport, hntmp, sizeof(hntmp) ) ;
                printf ("%d",sizeof(hntmp));
                 */

            break;
    }


    return 0;
}


void CVICALLBACK HN_clbrmenu (int menuBar, int menuItem, void *callbackData,
        int panel)
{
    DisplayPanel (HN_clbr_Handle);
}

int CVICALLBACK HN_clbr_close (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

            HidePanel (HN_clbr_Handle);

            break;
    }
    return 0;
}

int CVICALLBACK gshow (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

            DisplayPanel (HN_MGHandle)  ;

            break;
    }
    return 0;
}

int CVICALLBACK Step_Velocity (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    char M_enable[4] = "EN";
    char M_jog_command[6] = "MJOG";
    char M_stop[6] = "STOP";
    char M_disable[5] = "DIS";
    char M_op0[10] = "OPMODE 0";
    char hn_command[30];

    float hn_speed;
    float hn_time;
    int flag;
    int dr;
    int hn_num;
    //float tmp [] = {0,0,0,0,0,0,0};
    //int itmp[] = {0,0,0,0,0,0,0};
    double D;
    double t [40], tt[40];
    short int hn_len;
    //char tmp[30];
    int i =0;
    float hn_vel;


    strcat (M_enable,"\r");
    strcat(M_op0, "\r");
    strcat (M_disable,"\r");




    switch (event)
    {
        case EVENT_COMMIT:


                FlushInQ (M_comport);
                FlushOutQ(M_comport);
                ComWrt (M_comport, M_enable, strlen(M_enable));
                ComWrt (M_comport, M_op0, strlen(M_op0));

                GetCtrlVal (HN_TABHandle[4], TAB_STEP_NUM,&hn_num);
                GetCtrlVal (HN_TABHandle[4], TAB_STEP_VEL,&hn_vel);
                GetCtrlVal (HN_TABHandle[4], TAB_STEP_TIME,&hn_time);

                hn_time = 3;



                while (i<(2*hn_num))

                {
                    if (i==0) t[0]=Timer();
                i+=1;
                hn_vel *= -1;

                //SetCtrlVal(panelHandle, PANEL_Amin,i)   ;
                //SetCtrlVal(panelHandle, PANEL_VEL,hn_vel)   ;


                sprintf(hn_command, "J %.2f %.0f", hn_vel, 1000*hn_time)                ;

                strcat (hn_command,"\r");
                ComWrt (M_comport, hn_command, strlen(hn_command));

                D =   (double) hn_time;
                Delay (D+2);

                 ProcessSystemEvents ();

                t[i] = Timer();
                tt [i-1] =  t[i]-t[i-1];

            break;
    }
    return 0;
}

int CVICALLBACK testt (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
```

```
{
        int i=1;
        float hn_t1, hn_t2, hn_time,hn_t0;
        FILE *fp;
        char M_enable[4] = "EN";
        char M_op0[10] = "OPMODE 0";
        char M_stop [10] = "STOP";
        int hn_max;
        float hn_speed;
        char hn_command [20];
        int Ltmp;
        int hn_key=1;
        float hn_pos;
        char hn_pfb[50] = "PFB";
        char message[1000];
        strcat (hn_pfb,"/r") ;
        //fp = fopen (logFile, "w");


        switch (event)
        {
                case EVENT_COMMIT:

                        i=1;

                        GetCtrlVal (HN_TABHandle[5], TAB_COMP_COMPKEY,&hn_key) ;

                        if (strlen(logFile)<3)
                        {
                                DisplayPanel (HN_FEHandle);
                                SetCtrlVal(HN_TABHandle[5], TAB_COMP_COMPKEY,0) ;
                                SetCtrlVal(HN_TABHandle[5], TAB_COMP_LOG,0) ;

                                break;
                        }

                        fp = fopen (logFile, "a");

                        SetCtrlVal (HN_TABHandle[5], TAB_COMP_M, hn_key);

                        if (hn_key ==1)
                        {

                                DSN_Save_Vars();



                         fprintf (fp, "Compression Test\n\n# \tTime\tDisp\tForce\n\n");


                                FlushInQ (M_comport);
                                FlushOutQ(M_comport);

                                strcat (M_enable,"\r");
                                ComWrt (M_comport, M_enable, strlen(M_enable));

                                strcat(M_op0, "\r");
                                ComWrt (M_comport, M_op0, strlen(M_op0));


                                GetCtrlVal  (HN_TABHandle[5], TAB_COMP_MAX_LOAD, &hn_max);
                                GetCtrlVal  (HN_TABHandle[5], TAB_COMP_VEL, &hn_speed);


                                sprintf (hn_command, "J %1.2f",-1*(hn_speed));
                                strcat (hn_command,"\r");


                                hn_t1=Timer();
                                ComWrt (M_comport, hn_command, strlen (hn_command));

                                hn_t0=Timer();

                         for (i=1;;i++) {


                                        FlushInQ (LC_comport);
                                        FlushOutQ(LC_comport);

                                        QB_Read_LC();
                                        QB_Convert_FM();

                                        hn_t2 = Timer ()-hn_t0;
                                        hn_time = hn_t2;



                                        SetCtrlVal (HN_TABHandle[5], TAB_COMP_TT, LC_FM_loads[2]) ;
                                        fprintf
(fp,"%i\t%.2f\t%.4f\t%.2f\n",i,hn_time,0.001*(HN_position()),LC_FM_loads[2]) ;

                                        ComWrt (LC_comport, LC_stop, StringLength(LC_stop));


                                        if (-1*(LC_FM_loads[2])>hn_max)
                                        {

                                              strcat (M_stop,"\r");
                                              ComWrt (M_comport, M_stop, strlen(M_stop));

                                              SetCtrlVal(HN_TABHandle[5], TAB_COMP_COMPKEY,0) ;
                                              SetCtrlVal(HN_TABHandle[5], TAB_COMP_LOG,0) ;
                                              break;
                                        }

                                        GetCtrlVal (HN_TABHandle[5], TAB_COMP_COMPKEY,&hn_key) ;

                                        if (hn_key ==0)
                                        {

                                              strcat (M_stop,"\r");
                                              ComWrt (M_comport, M_stop, strlen(M_stop));


                                              SetCtrlVal(HN_TABHandle[5], TAB_COMP_LOG,0) ;

                                              fclose( fp );
                                              break;
                                        }

                                        ProcessSystemEvents ();
                             }

                            }


                            break;
      }


      return 0;
}

int CVICALLBACK HN_Cmpression_Test_File (int panel, int control, int event,
            void *callbackData, int eventData1, int eventData2)
{

      int hn_flag;

      switch (event)
      {
            case EVENT_COMMIT:

                  GetCtrlVal(HN_TABHandle[5], TAB_COMP_LOG,&hn_flag);

                  if (hn_flag){


                  FileSelectPopup ("", "*.log", "",
                              "Enter the name of the LOG FILE",
                              VAL_SAVE_BUTTON, 0, 0, 1, 0, logFile);}
                  //else logFile[] = { };

                  break;
```

```
        }
        return 0;
}


int CVICALLBACK CLOSE_ERROR_PANEL (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
        {
                case EVENT_COMMIT:

                        HidePanel (HN_FEHandle);

                        break;
        }
        return 0;
}

int CVICALLBACK HN_dim (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{

        char aa[100];
        int a1,a2,a3;
        char M_pos_feedback[5] = "PFB\r";

        switch (event)
        {
                case EVENT_COMMIT:




        FlushInQ (M_comport);
        FlushOutQ(M_comport);
        for (i=1;i<3;i++){
        //M_read_data[0] = '\0';
        ComWrt (M_comport, M_pos_feedback, strlen(M_pos_feedback));

        Delay (.02);
        //ComWrt (M_comport, M_pos_feedback, strlen(M_pos_feedback));

        //Delay (.02);
        a1 = GetInQLen(M_comport);
        M_bytes_read = ComRdTerm(M_comport, aa, a1, 10);

        a2 = atoi(aa);
        //Convert the position to an integer



        //Delay(1);
        }

        SetCtrlVal (panelHandle, PANEL_text2, a2);
        ProcessSystemEvents();


                        break;
        }
        return 0;
}


int CVICALLBACK DISABLE (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{

        char M_disable[5] = "DIS";
        strcat (M_disable,"\r");

        switch (event)
        {
                case EVENT_COMMIT:

                        ComWrt(M_comport,M_disable,strlen(M_disable));

                        break;
        }
        return 0;
}


//****************************************************
//  Function for reading and returning
//  the position of the actuator
//
//****************************************************

int HN_position (void)
{

        int a1, a2;
        char aa[30];
        char M_pos_feedback[5] = "PFB\r";


        FlushInQ (M_comport);
        FlushOutQ(M_comport);

        for (i=1;i<3;i++){

        ComWrt (M_comport, M_pos_feedback, strlen(M_pos_feedback));

        Delay (.02);

        a1 = GetInQLen(M_comport);

        M_bytes_read = ComRdTerm(M_comport, aa, a1, 10);

        a2 = atoi(aa);
        //Convert the position to an integer

        }

        ProcessSystemEvents();

        return a2;

}

int CVICALLBACK Camera_Start (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
        {
                case EVENT_COMMIT:

                        FlushInQ (CAM_comport);
                        FlushOutQ(CAM_comport);
                        ComWrt(CAM_comport,"S\n",2);

                        break;
        }
        return 0;
}

int CVICALLBACK Cmera_Stop (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
        {
                case EVENT_COMMIT:
                        FlushInQ (CAM_comport);
                        FlushOutQ(CAM_comport);
                        ComWrt(CAM_comport,"E\n",2);

                        break;
        }
        return 0;
}
```

IV. The permissions

There are eight figures in the thesis which are pulled out from other articles as mentioned in the footnotes. The permissions for Figure 4 which was taken from [35], Figure 5 which was taken from [38], and Figure 17 which was taken from [1] can be presented by request.