

DEVELOPMENT AND IMPLEMENTATION OF A LOW COST IMAGE
CORRELATION SYSTEM TO OBTAIN FULL-FIELD IN-PLANE
DISPLACEMENT AND STRAIN DATA

by

John Wesley Parker

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

of

Master of Science

in

Mechanical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

April, 2009

© Copyright

by

John Wesley Parker

2009

All Rights Reserved

APPROVAL

of a thesis submitted by

John Wesley Parker

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the Division of Graduate Education.

Dr. Douglas S. Cairns

Approved for the Department of Mechanical and Industrial Engineering

Dr. Chris M. Jenkins

Approved for the Division of Graduate Education

Dr. Carl A. Fox

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

John Wesley Parker

April, 2009

ACKNOWLEDGEMENTS

The author would like to give thanks to Dr. Cairns for his support and to Dr. McKittrick and Dr. Mandell for their knowledge and patience. He would also like to thank the Air Force Office of Scientific Research for their funding of this project. In addition, this project would not have been possible without the help of James Schmitt, Jay Smith, and Dan Samborsky. Finally, this thesis would never have been completed if not for Carol Anderson, Eileen Baker, Terry Kennedy, Joanne Wilke, Kayla Hunt, and Cheryl McNicholas. Their hard work and understanding is much appreciated.

TABLE OF CONTENTS

1. INTRODUCTION	1
Introduction	1
Composite Materials History	1
Composite Materials Today	3
2. COMPOSITE MATERIAL CHARACTERIZATION AND THE MSU IN PLANE LOADER	7
Composite Materials and Characterization.....	7
Dissipated Energy Density	10
The In Plane Loader	17
3. DIGITAL IMAGE CORRELATION SYSTEM	21
Digital Image Correlation Background and History in Mechanics	21
Cross Correlation	22
Sub-Pixel Techniques	25
Correlation Coefficient Peak-Fitting method	25
Gradient Method.....	26
Hardware System	29
The Camera.....	29
The Lens	34
Camera Mount.....	35
Camera Control system	37
Sample Preparation.....	38
Image Processing	39
Raw Image Format	39
Matlab applications.....	44
Overview of Functions	44
Control Point Grid Generation.....	46
Control Point Correlation	48
Boundary Displacements.....	51
Strain	53
4. TESTING THE ACCURACY AND PRECISION OF THE SYSTEM.....	57
Artificially Generated Translation and Strain.....	57
Artificially Displaced Images	57
Artificially Strained Images	59
Measuring Precision with a Stationary IPL Test	67

TABLE OF CONTENTS – CONTINUED

Uniaxial Tension on 6061-T651 Aluminum.....	70
Shear Test on a 6061-T651 Aluminum Dog bone.....	75
Summary of Testing Results	79
5. APPLICATIONS IN COMPOSITE MATERIALS TESTING	82
Boundary Displacements on Fiberglass/Epoxy Coupons	82
Boundary Displacements Glued Aluminum Tee-Joints	85
Boundary Displacements and Full-Field Strain on Carbon Fiber and Epoxy Coupons.....	89
Using DIC Strain for the DED function.....	93
6. IMPLICATIONS FOR COMPOSITE STRUCTURES CHARACTERIZA- TION.....	97
Non-Linear Constitutive Response	97
Full-Field Strain and Dissipated Energy	98
Other Implications	100
7. CONCLUSION AND FUTURE WORK.....	101
Future Work.....	101
Future Work, using the system for the non-linear constitutive response model and DED function.....	102
Future Work, improving IPL results with DIC	103
Improving the IPL itself	104
Conclusion.....	105
.....	105
REFERENCES CITED.....	106
APPENDICES	111
APPENDIX A: ASTM Extensometer report	112
APPENDIX B: Matlab Code	114

LIST OF TABLES

Table		Page
1	Hexcel IM7-G/8552 Prepreg Material Properties	13
2	Canon Digital Rebel 300d Sensor Properties	30
3	Image Translation Correlation Systematic Error for a Three Pixel Displacement	60
4	ASTM Classification of an Extensometer System.....	61
5	Stationary IPL Sample DIC Standard Deviations	69
6	6061 T6 Aluminum Properties from Two Tests.....	74
7	DIC Maximum and Minimum compared to FEM Results.....	79
8	Stiffness and R^2 Values	88

LIST OF FIGURES

Figure		Page
1	Ancient adobe kilns in Arizona. Photo by John P. O'Neill	2
2	Lockheed F-22	3
3	A one piece barrel section of the 787 Dreamliner fuselage. Using a one piece structure for the fuselage saves 40,000 to 50,000 fasteners.[10]	4
4	MIL HDBK 17 Building Block Approach for certification of primary composite structures. From Chapter 4 of the handbook.	6
5	The three material axes of a composite ply	8
6	Comparison of Tsai-Wu and Maximum Stress failure criteria to arbitrary experimental data	9
7	Dissipated Energy of a Carbon Composite during progressive failure	11
8	Carbon/epoxy IPL coupon geometry	14
9	The MSU In Plane Loader	14
10	Grip mounted LVDT system	19
11	Example of the cross correlation matrix with a near perfect correlation peak	25
12	A cutaway view of the canon Digital Rebel 300d [3].....	30
13	An example of CCD blooming on an image taken by the Faulkes Telescope in Hawaii [33]	31
14	A diagram representation of Canon's random noise reduction [13]	33
15	A cropped section of an ISO 12233 resolution test chart, units are in (line width/picture height)/100. This image was taken by the Canon 300d [4].	33
16	a Typical IPL test image, showing the camera field of view and a speckle painted carbon-fiber coupon	36
17	Camera, lens and camera mount on the IPL after testing an aluminum sample	36
18	Detail of speckle pattern at the notch of an IPL coupon	39
19	The Bayer filter color arrangement	41

LIST OF FIGURES – CONTINUED

Figure		Page
20	Bilinear color-space interpolation on a bayer filtered image	42
21	The DIC algorithm as implemented in MATLAB for this thesis	46
22	Two images from the boundary grid generating process on one corner of an IPL sample.....	47
23	A control point grid designed for the calculation of strain	49
24	Boundary displacements of a test with shear and tension showing output detail. The yellow marks are the center points of the blue linear curves and the blue marks are the control points as they were tracked.	52
25	3-D displacement array arrangement.....	54
26	Shear strain element showing the points used for the central difference approximation of shear strain.....	55
27	A plot of ϵ_{yy} on the surface of an aluminum dog bone.	56
28	Correlation Error for four Different Correlation methods for a 1 Pixel Translation	59
29	Linear Interpolation on a Row of Grey Values	60
30	Strain Error and Extensometer System Class	62
31	Measured Strain When Actual Strain = 0.25%	63
32	Measured Strain When Actual Strain = 8%	64
33	Artificially created ϵ_{xx} , ϵ_{yy} , and ϵ_{xy} strain	65
34	Measured Strain for 5% X Strain and 10% Y strain	65
35	Measured Strain for 5% Shear Strain	66
36	Fixed Error for Images With ϵ_x , ϵ_y and ϵ_{xy}	67
37	1 of 100 Images of a Stationary Sample in the IPL	68
38	An Aluminum Tension Test Sample.....	70
39	Modulus and 0.2% Offset Yield Strength of 6061-T651 Aluminum	71

LIST OF FIGURES – CONTINUED

Figure		Page
40	Stress vs. Strain of 6061-T651 Aluminum as Measured by VIC-2D and the MATLAB DIC Program.....	73
41	Modulus and 0.2% Offset Yield Strength of 6061-T651 Aluminum	74
42	6061-T651 Aluminum dog bone Geometry for use with the IPL.....	76
43	Boundary Displacements for a Shear Test on a 6061-T6 Aluminum Dog bone as reported by the DIC system.....	76
44	ϵ_y strain result from the ANSYS model. This figure also shows the element density.....	78
45	ϵ_y strain as calculated from the MATLAB DIC results.....	78
46	ϵ_y strain result from VIC-2D	78
47	ϵ_y strain as calculated from the DIC results.....	78
48	Boundary displacements in the y direction for a combination x and y test on a fiberglass/epoxy IPL coupon	84
49	Aluminum T joint. Gage length of 47 mm, width of 25 mm, glue area of 16.1 cm ²	85
50	Boundary displacements in the y direction for a y test of an aluminum T joint as reported by the original DIC scripts, the LVDT's and as predicted by ANSYS	87
51	Boundary displacements in the y direction for a y test of an aluminum T joint as reported by the original DIC scripts, the LVDT's and as predicted by ANSYS	88
52	Boundary displacements in the y direction for a y test of an aluminum T joint as reported by the second and third generation DIC scripts and as predicted by ANSYS	89
53	Boundary displacements in the x and y directions for a combined x and y test on a carbon/epoxy IPL coupon	91
54	Boundary displacements in the y direction for a y test on a carbon/epoxy IPL coupon.....	91
55	ϵ_y strain in a pure y displacement test just after damage initiation	93
56	ϵ_y strain in a pure y displacement test with growing crack	93

LIST OF FIGURES – CONTINUED

Figure		Page
57	Selected points in the notch area to indicate their position for positioning the strain-field in development of the DED function	94
58	ϵ_y and ϵ_x strain on an IPL sample and a bilinear constitutive response .	98
59	How the DIC system is used to develop the DED function using full field strain	99

ABSTRACT

When using multi-axial testing, obtaining in-plane strain and displacement data can be difficult and costly. Montana State University's In Plane Loader (IPL) can apply any number of loads in the X and Y axes and as rotation but can only provide the displacement data at the grips. This project uses a Canon consumer-model digital single lens reflex camera and a series of matlab codes to provide in-plane displacement and strain data at as many points on the surface of the loaded specimen as required using Digital Image Correlation. The system has been successfully used to measure surface and boundary displacements on composite and aluminum samples, as well as on glued aluminum joints. The software system has been shown to estimate strains with a strain error of less than 0.0002 m/m or $\pm 0.5\%$ placing it within ASTM-B2 classification. It also has been shown to measure displacements to within 0.02 pixels accuracy, which for tests on the IPL translates to 0.007mm. The hardware/software system has also been used with the IPL to estimate Young's modulus of aluminum with less than 5% error and yield strength with just over 10% error.

INTRODUCTION

Introduction

The purpose of this project was originally to develop a method for measuring accurate boundary displacements on composite coupons being tested in multiple planar load paths on the Montana State University In Plane Loader (IPL). It evolved into a portable image correlation system and procedure that can be used to measure not only boundary displacements but full field strain on samples being tested with the IPL as well as on other test machines. This thesis will begin with a brief history of composite materials and the difficulty of their characterization before delving into the workings of the IPL and the image correlation displacement measurement system and detailing how it can simplify the process.

Composite Materials History

An engineering composite material is a material made of two or more different materials with significantly different properties that remain separate on a macroscopic level. One of the earliest uses of a fiber reinforced composite material by man was in adobe bricks, as seen in Figure 1 on the following page, which have been around for thousands of years. Adobe typically uses a combination of clay and sand for the matrix and straw for the reinforcement. Adobe is a very durable building material and some of the oldest standing buildings in the world are made from adobe bricks. Adobe is still used today, primarily in hot climates because its large thermal mass keeps buildings cool in the hot sun.



Figure 1: Ancient adobe kilns in Arizona. Photo by John P. O'Neill

The composite materials referred to as such in this thesis are Fiber-Reinforced Polymers (FRPs), consisting of high strength reinforcement fibers embedded in a polymer matrix. The emergence of modern composite materials began in the 1930's with the large scale production of fiberglass by Owens Corning. One of the first fiber reinforced plastic structure was a boat hull made as an experiment in the mid 1930's. However, Fiberglass composite was not successfully used in a boat till polyester resins became available in the 1940's [2]. The advent of inexpensive FRP hulled boats made the sport accessible for more people and ushered in a new era in boating. Fiberglass fishing rods were another advancement around the same time that improved on bamboo rods in performance and decreased price drastically. While FRPs led to some major advances in the sporting goods industries, the largest strides were made in the aircraft industry. [2][47]

The use of FRPs in the aircraft industry was started in the early 1940's by the U.S. Navy and Air force. The high strength to weight ration and excellent corrosion resistance of the materials were seen as potentially be very useful in the corrosive salt water environment that is home to the Navy, although initial use was relegated to fiberglass and phenolic-resin composite tooling to stamp sheet metal into the complex

shapes needed for aircraft [45]. It was not until much later that FRPs saw significant structural use. The McDonnell Douglas F/A-18 Hornet was the first aircraft to be built with carbon fiber composite wings in the early 80's [9]. Since then, the use of composites in aircraft structures has steadily grown to where it is a primary structural component on many new planes.

Composite Materials Today

One of the first modern fighter jets to heavily use composite materials was the Lockheed F-22, shown in Figure 2, which is twenty-five percent composite materials by weight. It was originally projected to be fifty percent but the high cost of carbon fiber at the time and the perceived risk of the “new technology” kept it down [39]. This is changing however, as the new F-35 Joint Strike Fighter being produced by Lockheed Martin is almost forty percent composite materials [34].



Figure 2: Lockheed F-22

The most ambitious use of FRPs currently in aviation is in a commercial passenger jet. Boeing's 787 Dreamliner is constructed with fifty percent by weight FRPs consisting mostly of carbon/epoxy. The use of composites in the dreamliner has advantages far beyond just weight savings. Composite structural components are being integrated into the acoustical damping system, electrical system and thermal transfer system as well. The use of composites as the primary structure in the fuselage makes a much stronger structure then previous designs, allowing higher cabin pressure and larger windows which will make flying more enjoyable for the passengers. A small detail with a large affect that was made possible by FRP constructions was the elimination of a seam in the engine inlet. Eliminating this seam by combining the outer structure and inlet of the nacelle increases laminar flow enough to save an estimated 20,000 gallons a fuel a year[43].



Figure 3: A one piece barrel section of the 787 Dreamliner fuselage. Using a one piece structure for the fuselage saves 40,000 to 50,000 fasteners.[10]

Some other interesting new uses of FRPs are in unmanned air vehicles being developed by Boeing and Northrop Grumman. One such vehicle is the Boeing A160 “Hummingbird” which is a unmanned helicopter that is being designed to fly at altitudes of up to 28,000 feet or 18 hours with a 300 pound payload. It has already

flown for 12 hours with a 600 pound payload during testing [8]. Like the 787, it is largely composed of carbon fiber composites.

These advancements in the use of FRPs in commercial and military air structures will likely continue to carry through to other applications and drive the cost of consumer FRP products down even further, making them an increasingly more popular option for those that could not afford them previously. An example of this “trickle down technology” can currently be seen in the bicycle industry where carbon fiber composite components are no longer limited to only the most expensive products. Large manufacturers in China are producing low cost and high quality composite parts for many companies at prices that are lower than ever before.

One of the expensive processes thus far in designing with FRPs for military applications especially is in testing. MIL 17 specifications for testing use a multiple step technique. First small coupons of the composites are tested, proceeding to joints and other structures and finally moving on to entire assemblies as shown in Figure 4 on the following page. This approach involves many tests on many different geometries and assemblies and is very time consuming and expensive, with the primary expense being the many complicated and unique components that must be created and individually certified for each structure being developed. However, there are other methods for estimating the strength and post damage response of FRPs being developed that can be based on specimens that share the same geometry in combination with more advanced testing and well developed analysis. One such method being developed at MSU deals with the energy dissipated as damage accrues, discussed in detail in Chapter 3 on page 21. The basic idea of this method is to remove geometry from the certification process, thus removing the primary cost of the process. Tests on multiple components are replaced with multi-axial testing on small, inexpensive coupons that all share the same geometry, fiber orientation and

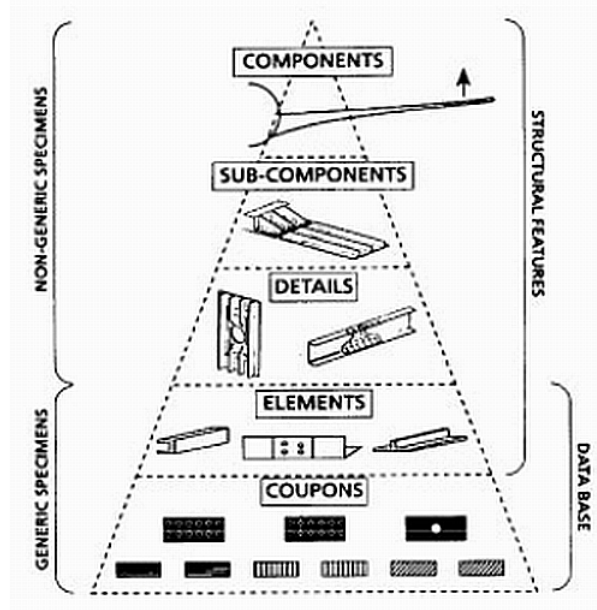


Figure 4: MIL HDBK 17 Building Block Approach for certification of primary composite structures. From Chapter 4 of the handbook.

processing. As long as the coupons are representative of the material and processing used for the final components, then the considerable costs of the upper level of the MIL 17 pyramid can be eliminated. Not only can the elastic constitutive properties of the laminate be determined from simple coupons but a non-linear response can also be modeled [40]. The Digital Image Correlation (DIC) system developed for this thesis further simplifies and improves on this process by removing the need to rely on finite element modeling of surface strains by measuring the actual strains during testing directly, thus eliminating a step and increasing the accuracy of the method. The MSU dissipated energy method, multiaxial testing the the DIC system will be discussed in detail in the following chapter.

COMPOSITE MATERIAL CHARACTERIZATION AND THE MSU IN PLANE LOADER

Composite Materials and Characterization

Characterization of composite materials provides some interesting challenges when compared to that of isotropic materials. Macroscopically isotropic materials such as metals and plastics have the same material properties in all directions. Characterization of such a material only requires determining two elastic constants and can be done with uni-axial tension tests. These materials also tend to display simple failure modes, typically a yield point at which deformation becomes permanent and an ultimate strength where failure occurs.[12]

Composite materials are anisotropic, meaning they have different properties in different directions. A fully anisotropic material requires 21 different elastic constants to fully describe its response to loading. The FRPs being discussed in this thesis are based on layers of continuous fiber composites with all the fibers in a layer oriented in one direction. Structures made of these layers will generally have multiple fiber directions but we will consider one layer for the sake of discussion. Continuous fiber FRPs can be considered to be orthotropic, meaning they have three planes of symmetry and three independent material directions as shown in Figure 5 on the following page.

These planes of symmetry allow the number of elastic constants that need to be determined to be reduced to nine. Further reduction to five constants can be made by assuming the material is transversely orthotropic, meaning the thickness and transverse directions behave the same [6].

Composite failure is also a complicated event. Composites don't have an

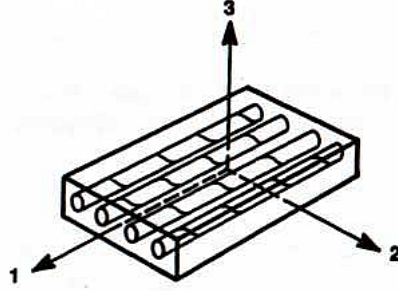


Figure 5: The three material axes of a composite ply

engineering “yield point” like more typical isotropic engineering materials; but instead a gradual accumulation of damage that eventually leads to a major failure. Damage can take many forms include fiber breakage, fiber debonding, matrix crazing and cracking and delamination [6]. This damage does not necessarily reduce the ultimate strength of the structure, but does affect its stiffness. These multiple forms of damage can lead to multiple modes of failure. Because of the complexity and multitude of factors involved, empirical criteria have been developed to predict failure. Empirical failure criteria are defined for the stresses acting on a single ply and generally define a failure envelope of combinations of shear and tensile/compression loads outside of which failure will occur. These formulas include Maximum Stress, Maximum Strain, Tsai-Hill and Tsai-wu[6].

Maximum stress and strain formulations define a simple rectangular stress or strain envelope with the maximum stress and strain values that a given ply can withstand. Tsai-Hill is similar to the Von Mises isotropic yield criterion only it is adapted to orthotropic materials by taking into account the interaction between the different failure strengths of the ply [42] [6]. The result is an ellipse that is more conservative for a ply being subjected to combination loading. The major drawback of using the Tsai-Hill criterion is that it cannot incorporate different behavior in tension and compression, which is important for polymer matrix composites. The Tsai-Wu

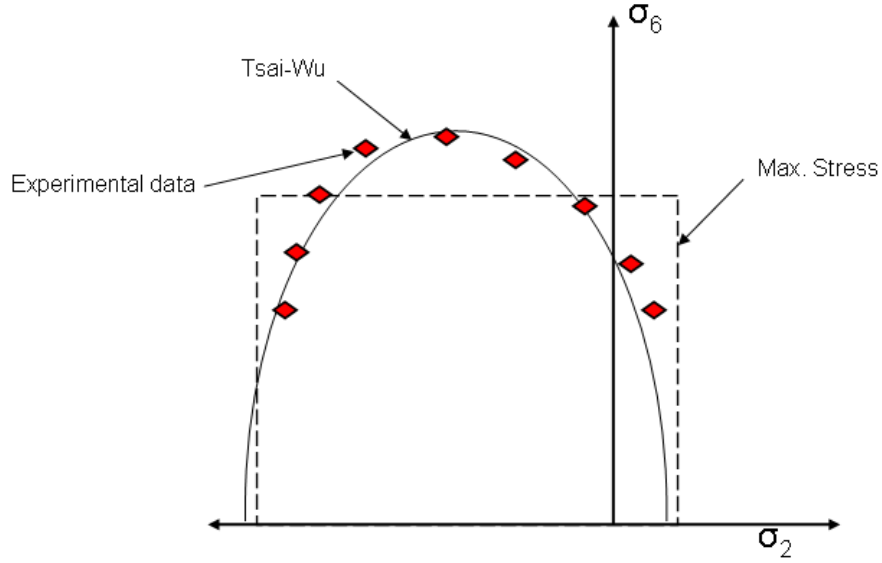


Figure 6: Comparison of Tsai-Wu and Maximum Stress failure criteria to arbitrary experimental data

criterion overcomes this by using a complete quadratic expression to attempt to fit experimental data. When written for an orthotropic material under it appears as follows in Equation 1.

$$F_1\sigma_1 + F_2\sigma_2 + F_{11}\sigma_1^2 + F_{22}\sigma_2^2 + 2F_{12}\sigma_1\sigma_2 + F_{66}\sigma_6^2 + F_{44}\sigma_4^2 + F_{55}\sigma_5^2 = 1 \quad (1)$$

With the tensile and compressive failure strengths in the 1 material axis written as σ_{1t} and σ_{1c} and in the 2 axis as σ_{2t} and σ_{21c} and with the shear failure strengths written as τ_{12} , τ_{23} , and τ_{31} the terms of the proceeding equation are defined as follows:

$$\begin{aligned} F_1 &= \frac{1}{\sigma_{1t} + \sigma_{1c}} \\ F_2 &= \frac{1}{\sigma_{2t} + \sigma_{2c}} \\ F_{11} &= \frac{1}{\sigma_{1t}\sigma_{1c}} \\ F_{22} &= \frac{1}{\sigma_{2t}\sigma_{2c}} \\ F_{66} &= \frac{1}{\tau_{12}^2} \end{aligned} \quad (2)$$

$$F_{44} = \frac{1}{\tau_{23}^2}$$

$$F_{55} = \frac{1}{\tau_{31}^2}$$

In equation 1 on the preceding page, σ_1 is the extensive force in the 1 material direction in the stress envelope and σ_2 is the compressive force in the 1 direction, σ_3 and σ_4 are the same in the 2 direction and σ_5 and σ_6 are in the 3 direction.

The final term F_{12} is defined in terms of the materials strength under equi-biaxial tension in the 12 plane σ_{b12} along with the previously defined terms and is written as:

$$F_{12} = \frac{1}{2\sigma_{b12}^2} [1 - \sigma_{b12}(F_1 + F_2) - \sigma_{b12}^2(F_{11} + F_{22})] \quad (3)$$

This term represents the interaction between the normal stress in the 1 direction and a normal stress in the 2 direction but not through shear. This term needs to be experimentally determined, generally through biaxial testing which is complicated and expensive[48][6][28].

In summary, the accepted biaxial strength theories are empirical formulations that require at best, a range of tests and specimen geometries to generate the required data. The criteria that most accurately fits experimental data, Tsai-Wu, requires biaxial testing which is difficult and expensive. The following technique, developed at MSU, using the In-Plane Loader and dissipated energy uses the same simple test specimens for all testing and can be used to quickly and inexpensively characterize a material.

Dissipated Energy Density

The Dissipated Energy Density (DED) technique has been developed along with the In Plane Loader (IPL) at Montana State University as a way to quickly, easily, and

cheaply characterize an anisotropic composite material. Dissipated energy density is used as a metric in this technique to represent damage to the material. DED is a proposed material parameter first used by Mast et al. [30] of the Naval Research Lab to quantify the amount of strain induced damage that has occurred in a composite material. The application of this technique has since been developed at MSU by Ritter [38], Smith [42] and most recently James Schmitt [40] for their Master's Theses. Dissipated energy can be easily visualized when looking at a force displacement diagram for a brittle composite that has started to behave non-linearly, as shown on actual IPL data in Figure 7.

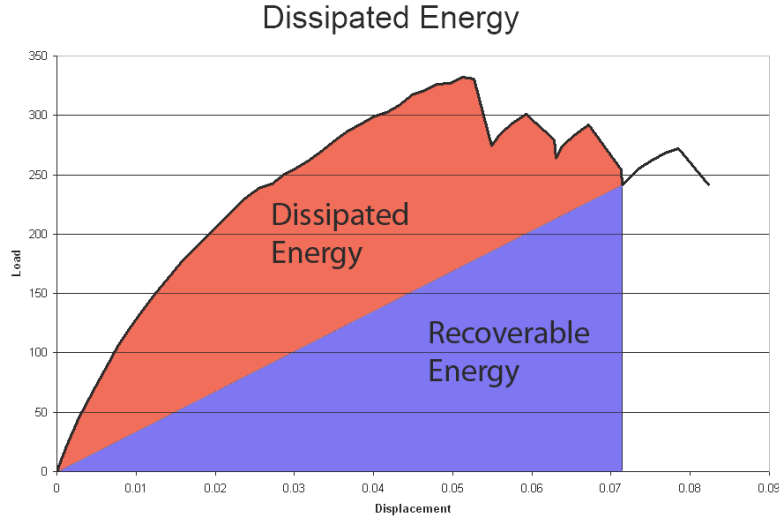


Figure 7: Dissipated Energy of a Carbon Composite during progressive failure

During such a test the total energy that has been applied into the sample can be measured as the area underneath the force displacement curve, represented as both the blue and red areas on the figure. The blue area represents the recoverable energy. If the load was released the displacement would return back to zero along the red/blue divide with no residual deformation but with a reduced stiffness. If the sample were to be loaded again its force-displacement curve would theoretically

pass linearly along this reduced stiffness curve until reaching it's previous value, then damage would begin to accumulate again. The dissipated energy is represented by the red area. It is the amount of energy that has gone into developing damage in the composite material. This damage reduces it's stiffness but does not manifest itself as plastic deformation as in a ductile material. It is proposed that dissipated energy in a composite is a scalar quantity that is only dependent on the material itself and it's strain state, and that a dissipated energy function can be developed that can predict the amount of damage in a structure of the same material based on it's strain state. The digital image correlation method used in this thesis ties into this technique by providing full-field strain data for the sample as well as boundary displacements, thereby increasing the amount of strain information that can be used to develop the DED function while cutting down on the amount of finite element modeling that has to be done. These improvements to the process will be covered in the ensuing step-by-step discussion.

The first step in the DED technique is to do a series of tests on a particular material with different multi-axial loadings. The goal is to subject the specimen to the largest variety of different strain states possible during all the tests in order to provide a very data rich environment for the development of a DED function. The most recent testing was completed by James Schmitt and John Parker in 2007-2008. More information on this testing and an analysis of the results can be found in Schmitt's thesis [40]. The material chosen for their tests was a IM7-G carbon fiber/8552 Epoxy resin unidirectional prepreg manufactured by Hexcel corporation. It is a high performance carbon fiber and toughened thermoset epoxy composite used in many aerospace applications. It's material properties are summarized in Table 1.

Layup was $[30/-30]_s$ and the laminate was cured under vacuum at 180 degrees celsius for two hours. The chosen coupon geometry was the same used successfully

Table 1: Hexcel IM7-G/8552 Prepreg Material Properties

Property	Value
Longitudinal Modulus, E_{11}	164.1GPa
Transverse Modulus, E_{22}	11.72MPa
Shearing Modulus, G_{12}	115.1MPPa
Major Poisson's Ratio, ν_{12}	0.3
Ply thickness (cured)	0.19mm

by Smith [42] and Ritter [38] in the past. It shares many of the same design characteristics of that used by the Naval Research Lab only larger and with a more conventional stress-concentration geometry. This geometry can be seen in Figure 8 on the following page. While it may look similar to that of a compact tension coupon used in determining fracture toughness, the notch serves somewhat different purposes, namely (1) Minimizing the effects of gripping the coupon by encouraging damage to happen as far from them as possible and (2) ensuring a large variation in strain over the surface of the coupon to increase the amount of information available for fitting the DED function to the data. A high contrast speckle pattern can be seen on the gage section of this coupon which has been applied for use with digital image correlation to get full-field displacement data. This process will be discussed in detail later.

The IPL is capable of producing three different boundary displacements either separately or concurrently. These are shear displacement X , vertical displacement Y and rotation R , as seen in Figure 9 on the next page. Testing is done in a series of load steps, the most recent testing used fifty steps per test although the number of steps before complete failure varied with the loading conditions. Loads were applied with the Y direction vertical as displayed in Figure 8 on the following page, the X direction horizontal and rotation about the center of the top boundary between the painted gauge length and the grip area. Seven unique loading paths were chosen in the most recent testing on the IM7/epoxy coupons. These paths were:

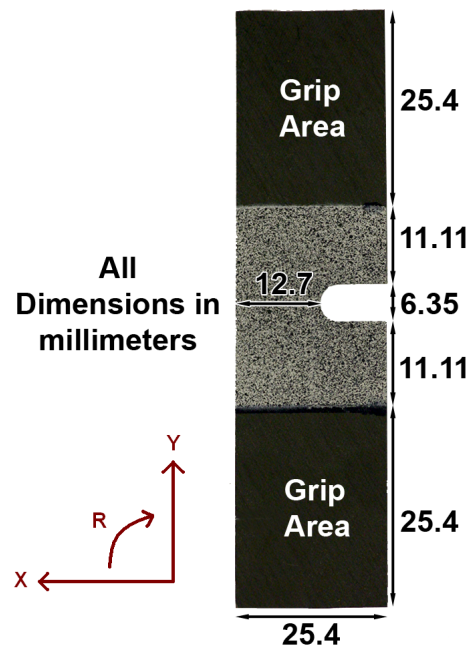


Figure 8: Carbon/epoxy IPL coupon geometry

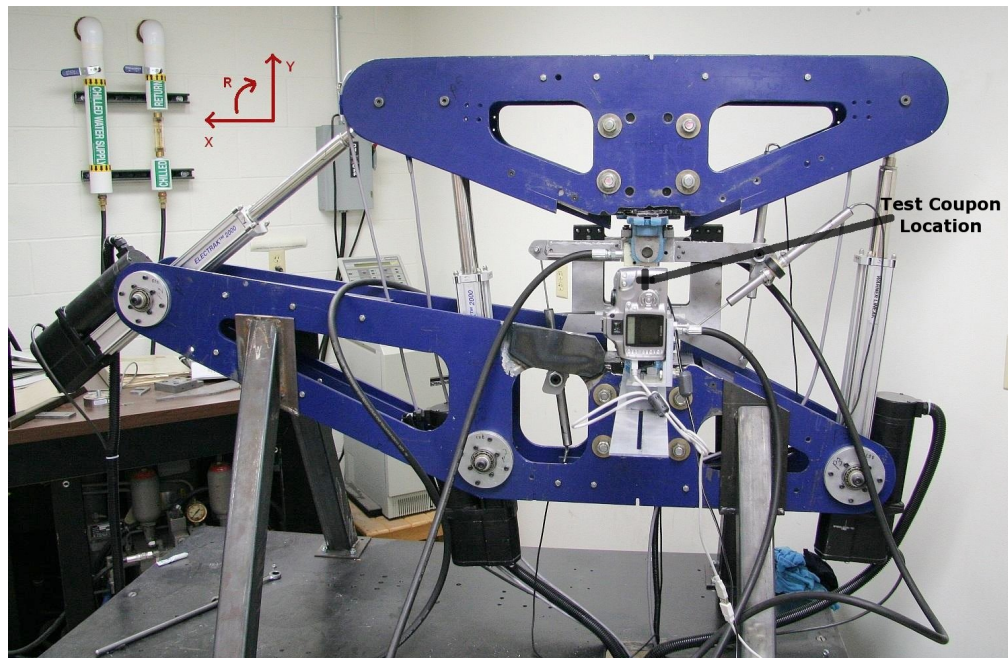


Figure 9: The MSU In Plane Loader

1. Pure tension, Y
2. Pure shear, X
3. Shear and Tension Y, X
4. Tension and Closing Rotation Y, R
5. Shear and Opening Rotation Y, R
6. Shear and Closing Rotation $X, -R$
7. Shear Tension and Opening Rotation $X, Y, -R$

These are similar to those chosen by Ritter and Smith in the past [38][42]. They were chosen to maximize the variety of the induced strains while avoiding large scale compressive forces so that buckling and out of plane displacements were not a problem.

Once load and displacement data is gathered for each test, the next step for finding the DED function is to acquire strain data from the coupon for each test. This can be done a couple of different ways. The technique used by Smith [42] and Ritter [38] in the past uses FEA models to effectively interpolate full field strain data based on the boundary displacements of the coupon. This has the advantages of not being affected by variations in material behavior over the gage length, but it does so at the expense of neglecting the actual stress-strain behaviour at any given point and replacing it with a modeled response. The DIC technique developed for this thesis allows strain data to be calculated over the surface of the sample directly from displacement data without using FEA. Using full-field displacement data for calculating strains has several benefits over using boundary displacements and a FE model to interpolate these strains. A notable advantage is that it allows

for displacement data to be gathered for any location desired. The only limitation to the number of strain points calculated by the DED technique is processing time. This allows for detailed strain information to be gathered from the areas with the largest strain gradients effectively increasing the amount of information available for the DED function. This method also adds some self consistency to the process of determining a constitutive response for a material from testing with the IPL that wasn't available before. This added consistency is because, when using DIC, the strain data used to develop the constitutive response and DED function comes from an actual laminate of the material. Some of the fiber-matrix and ply interaction that would be lost in an FE approximation are captured with the DED method. When the DED function is used to try and predict the behaviour of an actual structure it will ostensibly be more accurate having captured these interactions.

Dissipated energy is also calculated using the force data for the X , Y and R displacements for each test. These data and the strain data are the required results from testing for the DED method. The technique then becomes an optimization problem. The DED function is initially assumed to have the following form displayed as equation 4.

$$\phi(\bar{\varepsilon}) = \sum_i C_i N_i(\bar{\varepsilon}) \quad (4)$$

The strain vector $\bar{\varepsilon}$ is composed of the three in-plane strain components ε_{11} , ε_{22} and ε_{12} . These components are considered in ply-principle axes so the function is independent of layup. ϕ is the scalar quantity representing dissipated energy density previously determined from experimental data, N_i is a set of basis functions and C_i is a set of unknown coefficients to be determined from the experimental data that relate the strain data to the dissipated energy density. This equation is valid over a domain that extends to the extreme values of the experimental strain data. This

form of the equation was developed from that used by NRL [30], by Jay Smith, and James Schmitt [42]. Further information on the development of the DED function is beyond the scope of this thesis but can be found in Schmitt’s Master’s Thesis [40].

The In Plane Loader

The IPL was developed specifically for use with the DED technique of characterizing a composite material [23]. Rather than using several different sample fiber orientations, geometries and test fixtures with a uni-axial testing machine, the IPL allows an infinite number of in-plane loading paths to be applied to samples that share the same geometry and layup. Being able to use the same samples for all testing makes the characterization of a composite much faster, simpler and less expensive than other processes.

The IPL began as a senior design project for Eric Booth and Mark Schaff in 2001 [23]. It has since undergone many changes which were made by Will Ritter, Jeremy Kingman, Sarah Grochowski, Eric Booth, Bryan Bundy, Aaron Collet, Jay Smith and John Parker to increase it’s functionality and accuracy [38][17][42]. The IPL is designed to be an inexpensive and portable alternative to the in plane loader at the Naval Research Lab used in [30]. It consists of a load frame, linear actuator displacement system, load cell load acquisition system and Linear Variable Differential transformer (LVDT) based grip displacement acquisition system. It now also has an integrated mount and camera for using DIC to calculate full field displacements during testing.

The frame itself is made from 6.35 mm thick steel plate that sandwiches strategically placed steel spacers as well as the actuation and control systems. The frame has an upper and a lower section. The upper portion of the frame is

supported by the three actuators and during testing it moves while the lower section remains stationary. The actuators are located at three non-orthogonal angles and are controlled by a program constructed in LabView that determines the necessary actuator displacements to achieve the desired grip displacements by solving a series of vector-loop kinematic equations simultaneously for each movement[17]. More information on LabView can be found in [32].

The actuators used are Warner Electrak 2000 parallel ball-screw models [49]. Each actuator is rated to produce 9,000 Newtons of force. Maximum load ratings for the IPL have never been fully investigated but maximum tension has been found to be around 18,000 N, which is not surprising given that tension is produced entirely by the two most vertically positioned actuators. Samples for use in the IPL are generally designed to fail well below this limit to minimize the amount of frame deflection. Frame deflection has been a constant difficulty in using the IPL and has lead to the development of three separate systems for measuring displacement, the last of which is the optical DIC system that is the subject of this thesis.

The actuators are driven by stepper motors and their positions are measured in steps. For each actuator one step is equal to $0.5\mu m$, giving the IPL a very small position resolution limit that has never been reached during testing. Measuring displacements in terms of actuator steps was done with some success by Ritter [38] but problems with unwanted deflection were noted and mentioned as something that should be corrected in the future. Aaron Collett performed a validation study of the IPL in 2006 [17] and in the course of his study deemed frame deflection too large of a problem to work around in testing. He installed the second displacement measurement system which uses LVDT's mounted directly to the grips as seen in figure 17.

The three LVDTs in the system calculate displacement at the grips through vector

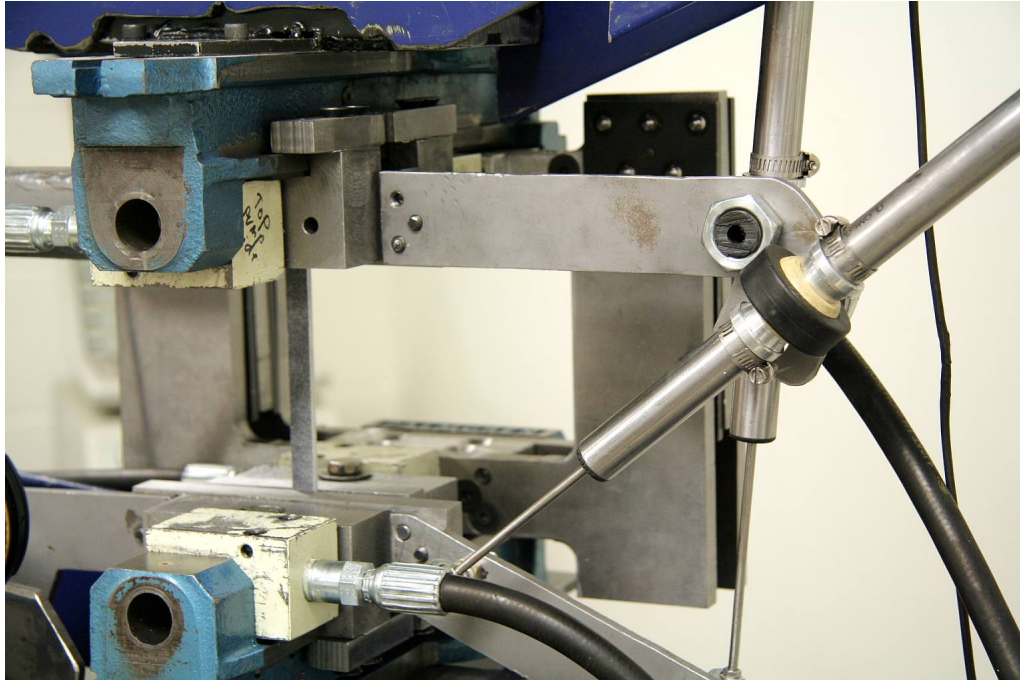


Figure 10: Grip mounted LVDT system

loop equations like those used for the actuators. They are used to correct the actuator displacements at each load step using a feedback loop until the grip displacement is within a prescribed convergence. This feedback loop adds a considerable amount of time to each test but compensates for the frame deflection of the IPL which can be quite large when testing stiff composite coupons.

Since Collett's LVDT displacement system was installed there have been two recent modifications to the gripping system completed by Jay Smith [42] and John Parker which include a new abrasive gripping surface and increased gripping pressure. The gripping surface is a carbide alloy abrasive surface that was metallurgically bonded to the grip surface by a company called Carbinite Metal Coatings. Two sets of grips were made, one set with a 100 grit coating and one set with a 60 grit coating. The 100 grit grips are currently in use and seem to work well on carbon/epoxy composites. The pressure of the gripping system was also increased to provide up to

55 MPa of gripping pressure. Despite these modifications, It was apparent that there was still undesirable coupon movement during testing, especially with tests involving rotation or shear in combination with testing. Getting accurate displacement data for tests done with the IPL was determined to be a priority and became the inspiration for this thesis. This led to the development and implementation of the current DIC system, which uses a consumer camera and lens combination to take images at each load step during an IPL test. These images can then be used to generate accurate boundary displacement data as well as full field displacement and strain data for use in general testing or the development of the DED function, the process for which will be discussed in the following chapter.

DIGITAL IMAGE CORRELATION SYSTEM

Digital Image Correlation Background and History in Mechanics

The basic premise of digital image correlation (DIC) as used in engineering mechanics is simple. A camera records images of a subject while it is undergoing some kind of deformation or movement. Each image is divided into numerous sub-images, each of which is centered on a correlation control point where displacement will be calculated. Sub-images are then displaced incrementally and compared to the previous images in the sequence using a correlation function until a maximum similarity is found. The displacement needed to achieve maximum correlation is taken to be the displacement of the control point and the process is repeated for all points and images.

The DIC process is generally divided into two different stages. The first stage takes discrete, one pixel steps and correlates each one. The highest correlation is then taken to be the starting point for the next stage, which uses interpolation for sub-pixel accuracy. This second stage has been done successfully by others using several different techniques of varying speed and accuracy. [50][41] Two of these techniques were developed in MATLAB for this thesis and will be discussed in great detail later.

The DIC method has seen widespread use in experimental mechanics as a way to measure displacements in mechanical testing as well as use in other fields such as biology for studying tissue displacements[51]. DIC was first used in mechanics to measure small strains in 1982 by W.H. Peters and W.H. Branson [50]. It has since developed into a popular way to measure displacements and strains where its high sensitivity and non-contact nature are beneficial. Very large and very small

applications are popular, as well as applications on complex geometries or in harsh environments.

The mathematics, hardware and software behind the DIC system developed for this thesis for measuring two dimensional displacements will now be discussed beginning with the basic principles behind image correlation.

Cross Correlation

Cross correlation, often called a “sliding dot product” is a measure of similarity between two functions. The cross correlation of two complex functions $f(t)$ and $g(t)$ is defined below in Equation 5.

$$f \star g \equiv \bar{f}(-t) * g(t). \quad (5)$$

where $\bar{f}(t)$ represents the complex conjugate of $f(t)$. The complex conjugate of f is used so that imaginary portions of it will contribute positively to the sum. The correlation reaches its maximum value when all the peaks and troughs of each function line up and contribute positively to the sum. In image correlation, a discrete version of cross correlation is used.

The DIC technique deals with images as discrete, real functions. Each pixel on the image has a numerical luminance value and the images are dealt with simply as arrays of these values. Image correlation between two images is done at user specified control points. For every control-point being tracked on the sample there is a larger base-image, $f(x, y)$, cropped from the previous image in the sequence and centered on the control-point and a sub-image, $t(x, y)$, cropped from the current image in the set. The sub-image is originally centered on the original location of the control-point being tracked from the base-image but is then moved over each pixel location on the base

image within defined search range and correlated to it. The basic cross correlation function for a two dimensional discrete case is quite simple can be written as shown below in Equation 6.

$$c(u, v) = \sum_{x,y} f(x, y)t(x - u, y - v) \quad (6)$$

In this case u and v represent the x and y pixel displacement being correlated. At each (u, v) pair, the correlation is calculated over a chosen range of (x, y) values from the two images that define the feature or contain the point being tracked.

There are a few reasons why this simple equation does not work for the DIC technique.

- This equation is sensitive to image intensity. It would be possible that the correlation between two bright spots on the image would be greater than where all of the features line up.
- The correlation value is also sensitive to changing lighting conditions over the sub-images.
- The value of the correlation function $c(u, v)$ increases with the size of the images being correlated.

These problems can be overcome by normalizing the sub-images and their corresponding footprint on the larger base images to unit length. This yields the equation for Normalized Cross Correlation shown below in Equation 7.

$$\gamma(u, v) = \frac{\sum_{x,y} [f(x, y) - \bar{f}_{u,v}] [t(x - u, y - v) - \bar{t}]}{\left\{ \sum_{x,y} [f(x, y) - \bar{f}_{u,v}]^2 \sum_{xy} [t(x - u, y - v) - \bar{t}]^2 \right\}^{0.5}} \quad (7)$$

where $t(x - u, y - v)$ is the sub-image after being displaced by u in the x direction and v in the y direction, $f(x, y)$ is the base-image, $\bar{t}$ is the mean of the sub-image and $\bar{f}(u, v)$ is the mean of the base image $f(x, y)$ in the region under the sub image.

This is the function that is most popularly used in image correlation and it was used in this thesis. By normalizing the components of the correlation function to unit length, sensitivity to image intensity, changing lighting conditions or image size is removed. This function will output a normalized correlation coefficient that varies from 0 to 1 with 1 indicating a perfect correlation.

The size of the sub and base images depend on several factors. They need to be large enough that the patterns on the surface are distinct enough to avoid false correlations, yet small enough to keep processing and search time reasonable for the application. Correlation sizes will be discussed in more detail later in this thesis. The result of running the normalized cross correlation function on a base-image and sub-image is a matrix of cross correlation values. An example of a surface plot of a matrix showing near perfect correlation for a 61x61 pixel base-image is shown in Figure 11 on the following page. The maximum normalized correlation value in this matrix is 0.994, which indicates near perfect correlation on a pixel level. There was no displacement between these particular images so that is not a surprise.

The maximum value on this matrix indicates the pixel location of maximum correlation between the sub-image and base-images. The precision of this measurement depends on the resolution of the image, which can be increased in two ways, *viz.*: (1) The resolution of the image sensor used can be increased and (2) the magnification of the image can be increased. There are drawbacks to each of these options. Increasing camera resolution rapidly increases expense and doesn't necessarily increase the image resolving power when using a consumer Digital Single-Lens-Reflex (DSLR) camera as was done in this thesis. Increasing the magnification can only be done to the near limit

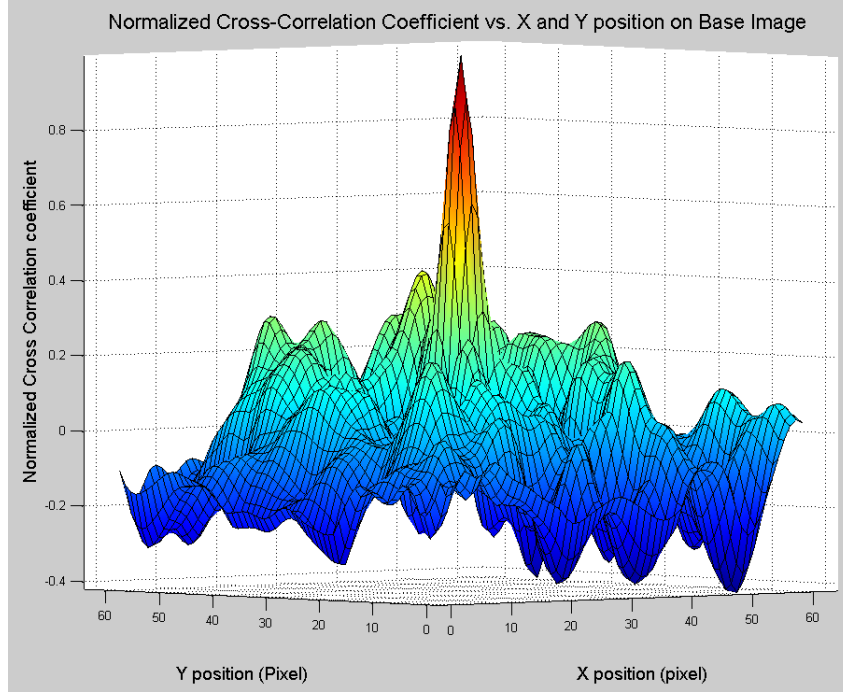


Figure 11: Example of the cross correlation matrix with a near perfect correlation peak

of the optics being used and it correspondingly decreases the field of view that can be analyzed. A third method for increasing resolution is to use a mathematical technique to gain sub pixel accuracy, of which there are several. Two of these techniques were investigated in detail in this thesis and will be covered in the following section.

Sub-Pixel Techniques

Correlation Coefficient Peak-Fitting method

There are multiple ways to achieve sub pixel accuracy that have been used successfully, several of which are outlined in [35],[41],[27], and [18]. The first technique used for this thesis is the least computationally intensive of the techniques found and will be referred to as the correlation-coefficient peak-fitting method. This technique starts with the matrix of normalized correlation coefficients from the previous step

and simply fits a second order polynomial to the nine pixel locations surrounding the peak of the correlation matrix. The equation for this surface is shown below as Equation 8.

$$C(x_i, y_j) = a_0 + a_1x_i + a_2y_j + a_3x_i^2 + a_4x_iy_j + a_5y_j^2 \quad (8)$$

The maximum value of this fitting surface will indicate the x and y location of the maximum correlation to sub pixel accuracy by essentially interpolating between the nine points in the three by three area surrounding the correlation peak. The maximum value can then be found by solving the following set of linear equations 9 and 10.

$$\frac{\partial C(x, y)}{\partial x} = a_1 + 2a_3x + a_4y = 0, \quad (9)$$

$$\frac{\partial C(x, y)}{\partial y} = a_2 + 2a_5y + a_4x = 0. \quad (10)$$

By solving equations 9 and 10 the interpolated value of the maximum correlation coefficient is calculated as shown in Equation 11.

$$x = \frac{2a_1a_5 - a_2a_4}{a_4^2 - 4a_3a_5}, \quad y = \frac{2a_1a_5 - a_2a_4}{a_4^2 - 4a_3a_5} \quad (11)$$

From here the u, v displacement of the sub image can be found just by subtracting the calculated x, y coordinates of the sub image center from the x, y coordinates of the undeformed image's subset center. This technique was used extensively for this thesis and generally provided satisfactory results. The second technique used, however, was determined to be more accurate.

Gradient Method

The second sub-pixel technique used will be referred to as the gradient based method. This method was originally developed by Davis and Freeman for use in video compression and is based on first-order spatio-temporal gradients. A quick explanation will follow, while detailed information can be found in [18] and [19].

Consecutive video frames generally contain a lot of data that remains relatively unchanged over time and can be removed as part of a compression algorithm. How unchanged over time the spatial image information is can be measured by calculating first order derivative between frames, referred to as the spatio-temporal gradient, and considering parts of the frame with a low enough rate of change stationary and viable for compression. It has been demonstrated in [19] that this technique is fundamentally the same as using an iterative block matching technique, common in basic image correlation algorithms, where the sub-image “block” is linearly interpolated by sub-pixel increments and correlated to the base-image. Block matching methods use an iterative search technique to adjust displacements until a maximum correlation is found. The gradient based method achieves the same results by re-casting the original block matching problem into a linear system with a closed form solution that has been shown to give the same sub-pixel results with a simpler implementation [18]. It was chosen for this project because of the simplicity of its implementation in MATLAB and the likelihood that it would provide faster processing times than a block matching algorithm.

This method starts with an assumption that the image misalignment is purely translational. This assumption is deemed to be reasonable when the sub-images being used and their displacements are small. With this assumption it can be assumed that the relationship in Equation 12 holds true.

$$f(x, y) = t(x + u + \Delta x, y + v + \Delta y), \quad (12)$$

where u and v represent the pixel value of maximum correlation and Δx and Δy are the sub-pixel displacement values. If the right hand side of this equation is expanded as a first-order Taylor expansion Equation 13 is the result.

$$t(x + u + \Delta x, y + v + \Delta y) = t(x + u, y + v) + \Delta x \cdot t_x(x + u, y + v) + \Delta y \cdot t_y(x + u, y + v), \quad (13)$$

where t_x and t_y are first-order partial derivatives of the sub-image intensities. These are approximated using a “five point rule” first order central difference formulation of order $O(h)^4$, where h is the pixel spacing. Using this method, an approximation of the the partial derivatives’ value at location $t(i, j)$ in the sub-image can be calculated as follows in Equations 14.

$$\begin{aligned} t_x(i, j) &= \frac{t(i-2, j) - 8t(i-1, j) + 8t(i+1, j) - t(i+2, j)}{12h} \\ t_y(i, j) &= \frac{g(i, j-2) - 8t(i, j-1) + 8t(i, j+1) - t(i+2, j)}{12h} \end{aligned} \quad (14)$$

The weights in this interpolation can be derived either by differentiating the Lagrange polynomial $p_4(x)$ as shown in [1] or with a fourth-degree Taylor expansion as found in [31]. The truncation error of this formulation is small at $O(h^4)$ although there is additional error stemming from the error in the displacements. If the displacement error at each point is e this additional error is equal to $\frac{3e}{2h}$. This error increases as grid spacing decreases, therefore a balance between this error, truncation error and strain detail must be found to achieve an optimum combination of low error and detailed strain information.

In practice this approximation is done simply by convolving the sub-image matrix with the mask $1/12[1 \ -8 \ 0 \ 8 \ -1]$ in the x direction to approximate t_x and then a transposed version of the mask in the y direction for approximating t_y . The size of the resulting matrix is then trimmed on all sides to eliminate values that were computed without the mask fully overlapping the sub-image matrix. Once partial derivatives are estimated equation 13 on the previous page becomes a set of linear equations that can be solved using the least squares method. This produces a closed form solution for the displacements that follows as Equation 15.

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sum \sum t_x^2 & \sum \sum (t_x \cdot t_y) \\ \sum \sum (t_x \cdot t_y) & \sum \sum t_y^2 \end{bmatrix}^{-1} \times \begin{bmatrix} \sum \sum [(f - t) \cdot t_x] \\ \sum \sum [(f - t) \cdot t_y] \end{bmatrix}, \quad (15)$$

where the sums are over all the x and y points contained in the discrete derivative matrices of t_x and t_y and the function t is evaluated including the pixel displacements as $t(x + u, y + v)$. The base-image and sub-image matrices used for this solution are trimmed to the same dimension of the partial derivative matrices before the solution is calculated.

The gradient method is only slightly more computationally intensive than the correlation coefficient peak fitting method and can improve results, which will be discussed in detail in a later chapter. With a review of the mathematics involved in the DIC technique out of the way, its implementation can now be discussed. The key component for the system is the camera.

Hardware System

The hardware used for this project was chosen with cost and practicality in mind. The hardware system includes the camera, lens, camera mount and control system. There are a wide variety of scientific cameras available but for this project a consumer digital single lens reflex (DSLR) camera was used to see if it could provide results that could compare to what has been done with scientific cameras. The camera chosen for this project was a Canon Digital Rebel 300d. More information on this camera can be found in a detailed review and benchmark at [3].

The Camera

The Digital Rebel was a revolutionary camera when it was first introduced in 2003. It was the first DSLR camera made by any manufacturer that sold for less than a thousand dollars. Despite its reasonable price it boasted exceptional image quality even when compared to more expensive cameras. It uses a large CMOS



Figure 12: A cutaway view of the canon Digital Rebel 300d [3]

(complementary metal oxide sensor) as its imaging sensor that exhibits lower noise than the sensors used in comparable cameras. Some specifications of this sensor are summarized in Table 2. A CMOS sensor differs fundamentally from the more

Table 2: Canon Digital Rebel 300d Sensor Properties

Property	Value
Sensor Resolution	3072x2048 pixels
Sensor Size	0.89 x 0.59 in. (22.7 x 15.1mm)
Sensor Aspect Ratio	3:2
Pixel size	7.4 microns

commonly known of CCD (Charge Couple Device) sensor. A CCD sensor contains a grid of metal oxide semiconductor photo-diodes that convert light energy into electrical charge. After this grid has been exposed to a light source it has to be read out. A full frame transfer CCD uses a serial readout register on one edge of the sensor and a single amplifier to amplify the signal into a usable form. When the image is read out, the first line of photo-diodes pass their charges onto the readout register as a complete row. The second line then passes its charge to the first row, the third to the second row and so on across the sensor. The values in the readout register are

then shifted in a perpendicular direction one at a time into an amplifier. The process is completed for each row of the sensor individually with charges being transferred from row to row toward the readout register across the sensor. There are several different ways that the readout architecture of a CCD can be arranged to increase shutter speed at the expense of cost and size but all of them are similar in that the pixel charges are fed one by one to the amplifier in the end. Some advantages to CCD sensors are a fill factor near 100 percent, meaning that almost all of their area can be used for gathering light, and overall lower systematic noise. Drawbacks to CCD sensors include the imaging artifacts of blooming, fading and smearing. Blooming is when a photo-diode is filled to capacity before it can be read out and energy spills over onto its neighbors. It is especially apparent in the long exposures of telescopes and can be seen on the two bright stars in Figure 13. This generally happens more

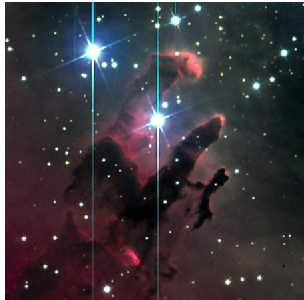


Figure 13: An example of CCD blooming on an image taken by the Faulkes Telescope in Hawaii [33]

easily in one direction then the other depending on the architecture of the CCD. Fading is when energy is lost as it is passed toward the readout register. Smearing occurs when light arrives at the sensor during the readout process.

In a CMOS imaging sensor each photo-diode is read out and amplified individually. This allows quick readout and eliminates fading and smearing. On a CMOS sensor pixels can also be read individually. Blooming can be prevented by reading out

pixels before they can overflow on an individual basis. There are drawbacks to CMOS architecture too however. For one, the readout and amplification circuitry takes up space on the surface of the chip that cuts down on the space available for catching photons. Reducing the size of this circuitry increases noise and manufacturing expense. Another problem is differences in amplification due to manufacturing tolerances in the amplification circuitry lead to a fixed pattern of variation in signal strength.

Canon has been developing CMOS technology for high end digital camera use since before the year 2000 and they have addressed these issues and others to improve image quality equivalent to that of CCD systems. A near 100 percent fill factor is achieved by mounting tiny micro-lenses above each pixel on the sensor to guide photons to the photoreceptors. Noise minimization is done at a hardware level and concentrates on two types of noise, *viz.*: pattern noise and random noise. Pattern noise is due to the differences in amplification between the per-pixel amplifiers as well as inconsistencies in structure and tolerances. These are compensated for by alternately sampling signal and noise from each individual photo-diode at 10 millisecond intervals and subtracting the noise from the signal each time. Random noise comes from multiple sources and includes shot noise, thermal noise and readout noise. Random noise is addressed on a per-pixel level by reading the charge induced from noise at each photo-diode separately from the charge induced by both light and noise and then subtracting the noise, as seen in Figure 14 on the following page [26][13]. Besides having low noise, the resolving power of the 300d is high relative to its low cost. Resolution of a camera and lens system is measured using a standardized ISO 12233 resolution chart and the units used are Line Width/Picture height (LW/PH). A crop from an image taken with the 300d of such a chart is shown in Figure 15 on the next page. The resolution on such a test is also dependant on the lens used, in this case it was a Canon 50mm prime

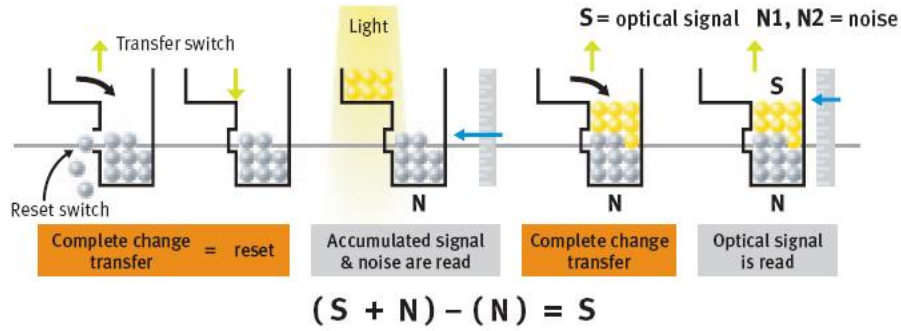


Figure 14: A diagram representation of Canon's random noise reduction [13]

that possesses resolving power well beyond that shown in the testing, indicating that the camera itself is the limiting factor.

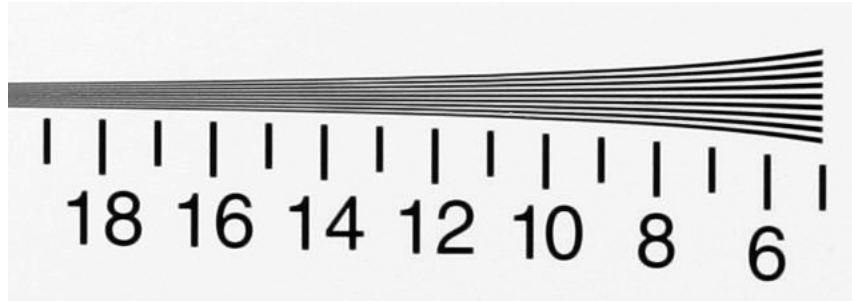


Figure 15: A cropped section of an ISO 12233 resolution test chart, units are in (line width/picture height)/100. This image was taken by the Canon 300d [4].

The maximum resolution can subjectively be determined with the naked eye as the point where the black lines can no longer be distinguished from each other. A more objective method uses the Modulation Transfer Function (MTF) which is the relative contrast at a given spatial frequency. The formulation of this is beyond the scope of this thesis but is done using additional portions of the ISO 12233 test chart. MTF is directly proportional to LW/PH.

While system resolution is dependant primarily on the lens and the cameras sensor, other factors also play a part. The 300d, like most consumer digital cameras, contains an optical low-pass filter attached to the front of its sensor to mitigate

aliasing and Moiré fringing on the images it produces. Moiré fringing occurs when an image contains spatial resolution that is above the resolving power of the imaging sensor. It is manifested in images by colored banding around the high detail regions. Low-pass filters, often referred to as anti-aliasing filters, slightly blur the image and block the higher spatial frequencies from hitting the sensor. The strength of this filter is determined using the Nyquist-Shannon Sampling theory where the sampling frequency (sensor resolution) needs to be twice that of the signal being sampled (the image). Canon’s method of reducing this is to use two layers of a birefringent crystal material with a phase plate sandwiched between them that changes linear polarization to circular [14]. The first crystal layer splits the light into two parallel beams, the second splits these two into four. This very precisely blurs the image over a small range of photoreceptors on the sensor, reducing resolution but removing Moiré patterns from the images produced. Moiré problems are also exasperated by the Bayer filter design of the sensor, which will be covered later.

In short, the camera chosen provides high resolution and low noise at low cost, and was deemed suitable for integration into the DIC system for this thesis. The next important component needed for the system was a suitable lens.

The Lens

The lens chosen for this project was the Tokina AT-X 100mm F/2.8 macro. Detailed information and optical testing results for this lens can be found in [36]. It was chosen out of several 100mm options because of its consistent edge to edge resolution, tested in [36] as 2073 LW/PH in the center of the image and 2053.5 LW/PH on the outer edge. Both of these are well above the resolving power of the camera’s sensor so therefore should not limit the system resolution significantly. A long 100mm focal length was chosen because it would allow the lens to be placed far

enough back from the sample to allow room for the grips of the IPL, and also make it easier to position for other testing applications. A macro lens was used because it was necessary to be able to fill as much of the field of view as possible with the small, 29mm tall IPL specimens. The IPL coupon geometry shown previously in Figure 8 on page 14. The angle of view of the Tokina lens is 23.3 degrees when used on a 35mm film camera or a full frame DSLR. The sensor on the 300d is smaller than the size of 35mm film with a crop factor of 1.6 which effectively reduces the field of view by a factor of 1.6 to 14.56 degrees. At the minimum focus distance of the lens this provides a vertical field of view of approximately 76mm at the sample with the horizontal field of view approximately 51mm. This is the minimum height and width that the lens can focus on and defines the smallest sample size that can be imaged at full resolution on the cameras sensor. It is also smaller than the commonly used IPL coupons and will therefore not be a limitation when used for them. A typical test image of a carbon IPL coupon can be seen in Figure 16 on the next page.

All samples were imaged the same, and while the lens was capable of fully filling the field of view with the width of the sample, extra space was imaged on either side to allow for large shear displacements to be used in testing without moving any part of the sample outside the field of view.

With a suitable lens, a mounting system had to be designed and constructed that would rigidly attach the camera to the IPL for testing.

Camera Mount

A mounting system was designed and built out of 6061-T6 aluminum as shown in Figure 17 on the following page. The system mounts directly to the bottom section of the IPL and holds the camera and lens either horizontally or vertically oriented at distances ranging from 0.3 to 0.6 meters. It was designed to rigidly hold the camera

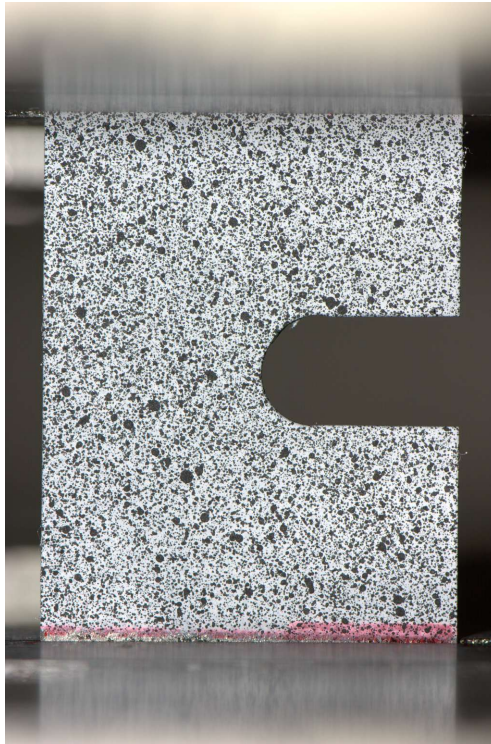


Figure 16: a Typical IPL test image, showing the camera field of view and a speckle painted carbon-fiber coupon

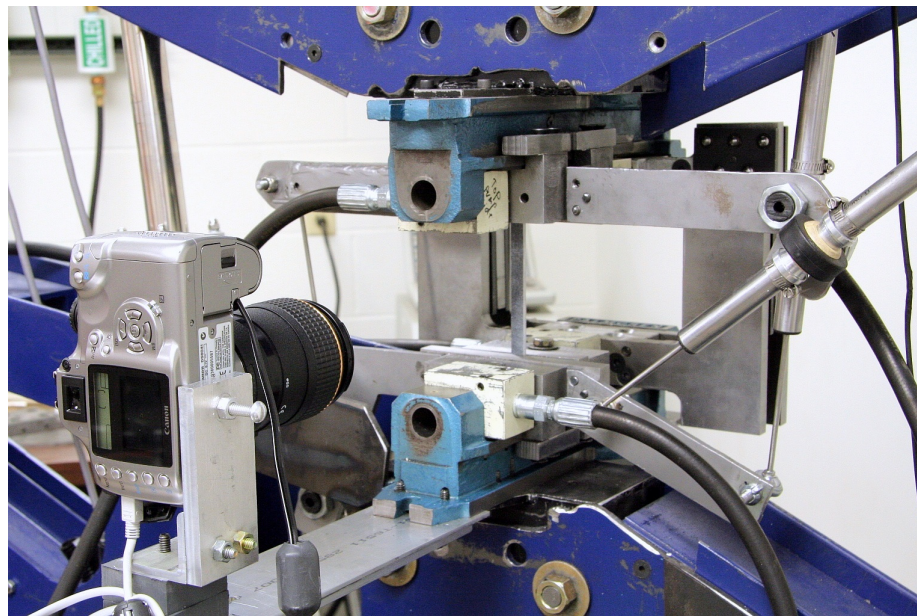


Figure 17: Camera, lens and camera mount on the IPL after testing an aluminum sample

lens perpendicular to the sample during testing. An attachment point was chosen as close to the grips as possible in order to avoid including the frame deflection of the IPL in the images. This was not a critical factor, as the bottom grip deflection could be subtracted from the data but it made for easier image processing and allows for a tighter field of view.

Camera Control system

The camera is directly controlled by the same computer as the IPL during testing. The IPL runs its tests as a series of discrete, displacement controlled steps. Most of the tests done in this thesis used 50 steps to achieve a good balance of data resolution and test length. At the end of each step the camera was triggered to take a picture after the IPL frame stopped moving. This was done using a very simple command line program from the Canon Software Developers Kit [15] that was run on a command line through LabView. Images were directly saved onto the camera's memory card and retrieved after the test for image processing on a separate computer. It was possible to download images directly to the control computer during testing but for the applications in the thesis this was never necessary as the image correlation was performed on a separate computer. For a reason that remains unknown the Canon control software added about a one minute wait to each step of the tests after taking an image. For IPL testing this was not very important since the LVDT displacement correction at each step takes a similar amount of time. However, for other uses or for IPL use without LVDT correction this extra time could be a hindrance. During the development of this system a trial of a third party software produced by Breeze Systems called DSLR Remote Pro [11] was tried and it provided a much faster turn

around time between images. If more speed is needed for future use of the system purchasing this program would be a good way to gain it.

For the testing done on the IPL for this thesis, the images were taken at 1/200 of a second at f/8.0 and ISO 100 using the on camera flash for illumination. The shutter speed is the maximum speed the camera supports for flash synchronization. The ISO setting relates to the amplification of the signal that occurs when the sensor is read out, so keeping this as low as possible reduces the readout noise. The aperture of f/8.0 was determined by the shutter speed and illumination but it is very close to the aperture that provides optimum lens resolution based on resolution charts from [36]. The on camera flash was found to provide excellent and even illumination for the sample without any problems with reflections. Focus was set by allowing the camera to auto focus once and then turning the focus to manual so it would not change over the course of the tests.

Sample Preparation

In order for the DIC technique to work, some kind of highly detail and high contrast pattern must be applied to the samples. For the present work this was done in the way recommended by several commercial DIC systems: with spray paint. A coat of non-reflective flat white paint was first applied as a base coat. Once this dried, a light coat of non-reflective flat black was overlaid leaving the speckle pattern shown below in detail in Figure 18 on the next page.

One difficulty with this preparation technique was the paint would sometimes flake off during large strains or after high energy damage events. To help prevent this the samples were prepped by sanding with 400 grit sandpaper and then cleaned with

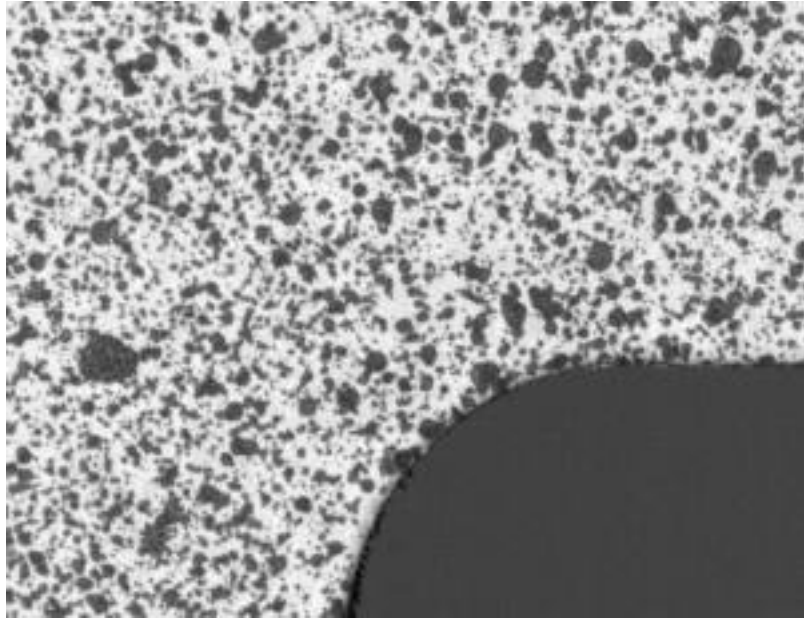


Figure 18: Detail of speckle pattern at the notch of an IPL coupon

isopropyl alcohol to give the paint more surface area to adhere to and remove any oils on the surface.

The speckle patterns that were used for this thesis varied somewhat as they were applied by hand. Some of them seemed to work better for DIC then others but this was never investigated fully. There is research available on this subject and it may be a suitable issue to investigate in the future. It would also be useful to find more flexible paint that would withstand more strain before flaking off and thus increase the amount of useful data available from testing.

Image Processing

Raw Image Format

Images taken by consumer DSLR cameras are generally stored in one of two formats, either as a “RAW” image or in JPEG format. JPEG stands for “Joint Photographic Experts Group” which is the name of the committee that established

it as a standard image format in 1991. JPEG images are 8 bits per channel, with 256 shades of red, green and blue per pixel. The total image can contain 8^3 different colors, which is close to 17 million and is a level of color detail referred to as “true color”. The JPEG format is a “lossy” format as opposed to the lossless compression used in RAW images. It accomplishes reduced file size by reducing the detail of the image in areas that are not easily perceived by human viewers. Encoding an image into the jpeg format takes place in several steps: [20]

- The color-space of the image is first changed to YCrCb which contains the luminance (brightness) as the Y channel and the chrominance (color) as the two channels Cr (red) and Cb (blue). The two chrominance channels are then immediately down-sampled, generally by a factor of two. The human eye is much more sensitive to luminance than chrominance because of a higher density of luminance sensing rods on the retina so this is barely noticed by the viewer.
- The image is then divided into 8 by 8 pixel blocks and a discrete cosine transform is performed on them. The frequency components of the transforms are then quantized. Low frequency components are stored with more detail than the higher frequency components since they are perceived more easily by the human eye. Components that are close to zero are discarded to further reduce file size.
- After quantization and elimination of zero values the data is then encoded using a lossless compression algorithm based on Huffman coding.

The result of JPEG encoding is images that don’t appear to be missing any information, but that is just because it has been stripped off in ways that are tough to perceive with the human eye and brain. In digital image correlation it is important to have as much information as possible stored in the image because every detail will

help when it comes time to calculate sub-pixel displacements. For that reason the RAW image format was chosen.

RAW format is a generalized term that encompasses several different propriety formats created by different manufacturers. The shared concept among all of them is that a RAW file is the raw output of the cameras sensor without any manipulation done by the camera itself stored in a proprietary way. Camera settings like exposure and color balance are included with the image data for the sake of being read when the file is “developed” by a RAW development program but the data itself is untouched. The 300d produces raw files recorded with 12 bits per pixel detail. This allows for 4095 shades per channel as opposed to the 256 in 8 bit images, an immediate increase in detail.

The major step in processing a RAW file stems from the way that color images are created in a consumer camera. The imaging sensor in the 300d, and almost all other available consumer cameras, is monochrome. Each pixel on the sensor is covered by either a red, blue or green filter. These filters are arranged in a repeating pattern known as a Bayer pattern, shown in Figure 19. This pattern was developed by Bryce

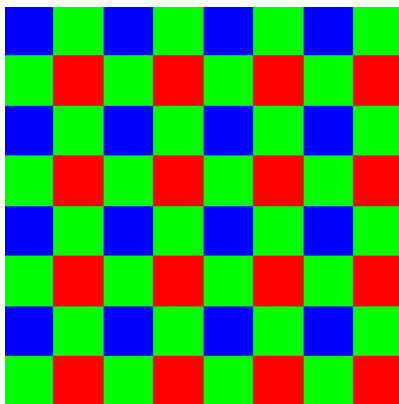


Figure 19: The Bayer filter color arrangement

E. Bayer of Eastman Kodak in 1975 [7]. It uses twice as many green filters as red

and blue. There are more green pixels than red or blue because the rods in the human eye are more sensitive to the luminance of green. As was mentioned before in the section on JPEG images, the human eye is also more sensitive to luminance than chrominance so increasing the number of pixels that can record luminance will directly increase perceived image quality. Most imaging sensors in single-sensor color cameras use the Bayer pattern, although there are a handful of other patterns that are also used, including a new one developed by Eastman Kodak that uses unfiltered pixels to increase the amount of light that the sensor can receive to increase the signal to noise ratio and effectively reduce noise in the image [21]. There is also a sensor made by Foveon that takes advantage of the fact that silicon absorbs red, blue, and green light to different depths to capture red, green, and blue at every pixel location with layered photo-receptors [24].

In order to generate color information from a Bayer filtered sensor, for the missing two channels at each pixel, some form of interpolation-based demosaicing must be done. One of the simpler methods for doing this is linear interpolation, shown in Figure 20. In this figure, the green value $G5$ would be calculated as the average of

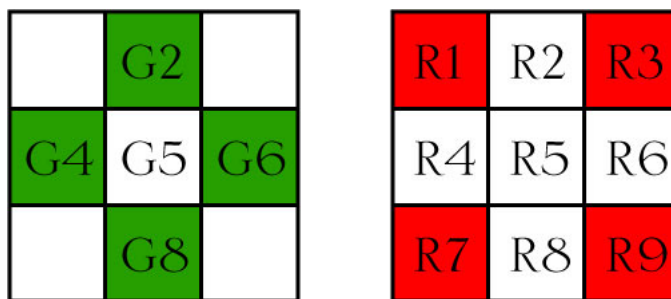


Figure 20: Bilinear color-space interpolation on a bayer filtered image

$G2$, $G8$, $G4$ and $G6$ The red value $R5$ would be calculated as the average of $R1$, $R3$, $R7$, $R9$ while $R8$ would simply be the average of the two values $R7$ and $R9$. Blue values would be calculated in a similar manner to the red ones. This technique

works well with smooth gradients but produced unacceptable color artifacts on edges with sharp color transitions. A common way to eliminate these is to remove large transitions in hue, with hue being defined as the ratios of Red/Green and Blue/Green, often with a median filter. While detailed transitions of luminance are common in images, hue transitions are generally more gradual so limiting them will not have much affect on the detail of the image, at least as far as the human eye can perceive [37]. By nature, all demosaicing algorithms will have some kind of error that is introduced in the resulting image as they interpolate and adjust color information for pixels that previously had none. A comparison of several interpolation algorithms is done in [37] that tabulates the interpolation error of several different techniques.

The RAW conversion for this thesis was done using the program DCRAW [16], which is an open source RAW decoder written by David Coffin. It was chosen because of the flexibility it offered in how the RAW images were processed. The method used for this thesis takes four pixel square blocks and combines their color information into a single pixel value. The four pixel blocks contain two green pixels, a red and a blue and are the smallest repeating portion of the Bayer pattern. The resulting image is one half the size of the original but there is no interpolation done that could affect the accuracy of sub pixel techniques.

Another processing technique, using DCRAW, that does not involve interpolation was also investigated. This technique uses a luminance multiplier for the red and blue filtered pixels in an attempt to match the luminance of the green. This method was designed as a quick way to process images of black and white text and is referred to as document mode in DCRAW. It was implemented with the auto white balance option to determine the multiplier, and the results were checked by numerically ensuring there was consistent luminance across the evenly colored background of the test images. This technique was compared in 1-D strain to the previous technique and the

difference between the two was found to be negligible. Since the images were much larger and processing time and storage was an issue it was not pursued further.

After RAW conversion the black and white images were then converted from 12-bits to 8-bits and imported into MATLAB for further processing. While 8 bit images lack the detail of the original 12 bit files, it was found in this thesis that using the 12-bit files had no discernible effect on the results but increased file size and processing time greatly.

Matlab applications

The programming in this thesis was done almost entirely in MATLAB due to availability at MSU and the availability of built-in functions that could be adapted to the DIC technique. Information on MATLAB and its image processing toolkit, which was used heavily in this thesis, can be found in [46]. Several scripts developed at John Hopkins University by Cristoph Eberl, Robert Thompson et.al. [22] for digital image correlation and tracking in MATLAB were used, as were several built in MATLAB functions from the Image Processing Toolbox and many other functions that were written for this thesis. In order to provide a reference for further discussion, the most important functions and scripts will be outlined below in the order that they are used in the image correlation process. The functionality of important functions will be discussed in detail later. The code used in this thesis is also included in its entirety in the Appendix.

Overview of Functions

- **JP_IPL_strain_grid_generator.m**- This function is used to build a grid of x and y control points that the DIC process will be carried out on. It gives

the option of defining a row of points at each sample boundary for the sake of generating boundary displacement data, or creating an evenly spaced grid over the sample surface for the sake of calculating full field strains. Its output is two files, `grid_x.dat` and `grid_y.dat`. When used with images, with the IPL sample geometry, it also stores the location of the notch in the sample as well as sample width in pixels.

- **filelist_generator.m** - This function was developed by Eberl [22] and simply generates a list of the image files that need to be correlated and saves it as `filelist.dat` to be read later.
- **automate_image.m** - This function was developed by Eberl [22] and modified for this thesis. It reads `filelist.dat` along with the `grid_y.dat` and `grid_x.dat` and correlates all of the control points in all of the images in succession. It uses the MATLAB's function `cpcorr.m` to do the pixel level correlation and then generates sub pixel displacement data using either the MATLAB function `findpeak.m` for correlation coefficient curve fitting or a modified version of the function `cpcorr.m`, named `cpcorr_gradient.m`, which was developed for this thesis and uses the gradient based sub pixel method. It outputs the files `validx.dat` and `validy.dat` which contain x and y pixel locations as they were tracked by the image correlation.
- **JP_boundary_displacements.m** - This script is run if a boundary displacement grid was used. It reads the `validx.dat` and `validy.dat` files, which in this case contain displacements for a row of control points at the top of the sample and another at the bottom. It produces x and y displacements and rotation data centered on the top and bottom boundaries and outputs these as the file `displacements.dat` for further use.

- **JP_inplanestrains.m** - This script is run if a strain grid was used. It reads the pixel displacements in validx.dat and validy.dat and generates ϵ_{xx} , ϵ_{yy} and ϵ_{xy} strain data. It outputs them as three-dimensional arrays in MATLAB with the dimensions of x grid point, y grid point, and image number.

The processing algorithm using these scripts is quite straightforward and is shown below in Figure 21

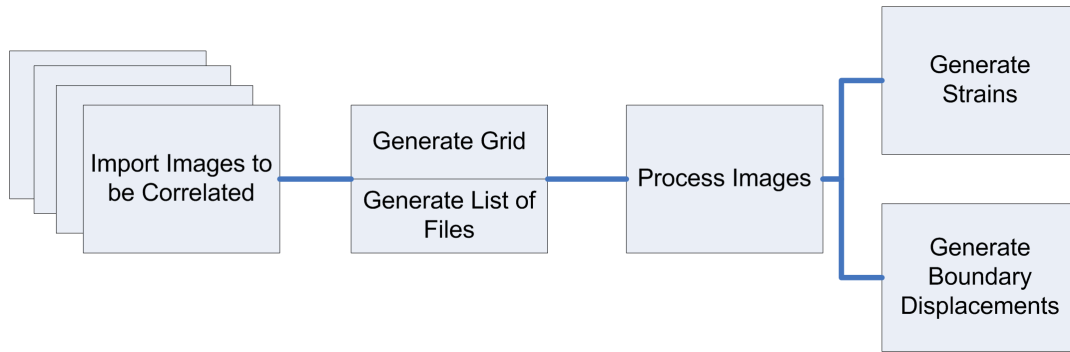


Figure 21: The DIC algorithm as implemented in MATLAB for this thesis

Control Point Grid Generation

The workings of the previous scripts and functions will now be discussed in more detail in the order in which they are applied. The first use of the DIC technique for this thesis was to find boundary displacements of IPL samples during testing. This started with using the JP_IPL_strain_grid_generator.m program developed for this thesis to create a grid of control points at the boundaries of the sample, as seen in Figure 22 on the next page. This program was designed to create evenly spaced rows of control points on the sample boundaries to be correlated and also to generate a conversion ration between the pixel displacements dealt with in the DIC technique and an absolute distance unit chosen by the user. After running the program the user is prompted to select four points on either side of the top and bottom boundaries as

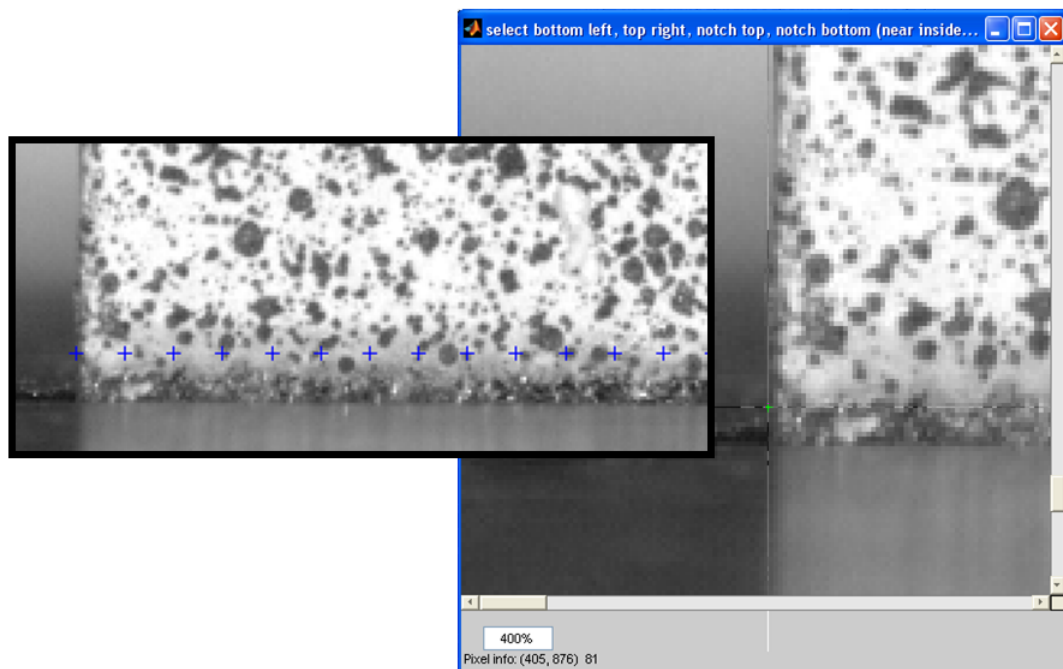


Figure 22: Two images from the boundary grid generating process on one corner of an IPL sample

accurately as possible. The dialog allows the user to zoom in on the corners to increase the accuracy of their placement. For the IPL samples the sample width was 2.54cm which was approximately 700 pixels on the images. It is quite easy to pick each edge of the sample to within a pixel of accuracy, which for the sake of creating a conversion factor from pixels to absolute displacement introduces a maximum error of approximately 0.14% as shown below in Equation 16.

$$\begin{aligned}\Phi_c &= \frac{25.4 \text{ in}}{699 \text{ pixels}} = 0.03633 \frac{\text{pixels}}{\text{mm}} \\ \Phi_c &= \frac{25.4 \text{ in}}{700 \text{ pixels}} = 0.03629 \frac{\text{pixels}}{\text{mm}} \\ \text{difference} &= \frac{0.03633 - 0.03629}{0.03629} \times 100 \text{ percent} = 0.143 \text{ percent difference}\end{aligned}\tag{16}$$

After the four edges are selected the boundary grid is evenly spaced across the top and bottom boundaries as can be seen in the previous figure. The program stores the upper and lower pixel widths of the sample after being run, which are later retrieved by the JP_boundary_displacements.m program as will be discussed later.

JP_IPL_strain_grid_generator.m can also be used to create a grid for use in calculating full-field strain information. It creates an evenly spaced grid over the whole sample surface, which can be seen below in Figure 23 on the following page. The grid is created between the four corners selected by the user. The points overlapping the notch on the sample are left in place at this point. It is easier to disregard the data produced inside the notch area during processing then to remove the points and compromise the even spacing of the grid.

Control Point Correlation

Once the grid of control points has been created it is time to process the images. After capture and RAW conversion the images are saved as 8 bit .tif images. These images are imported into MATLAB as matrices with each element containing an

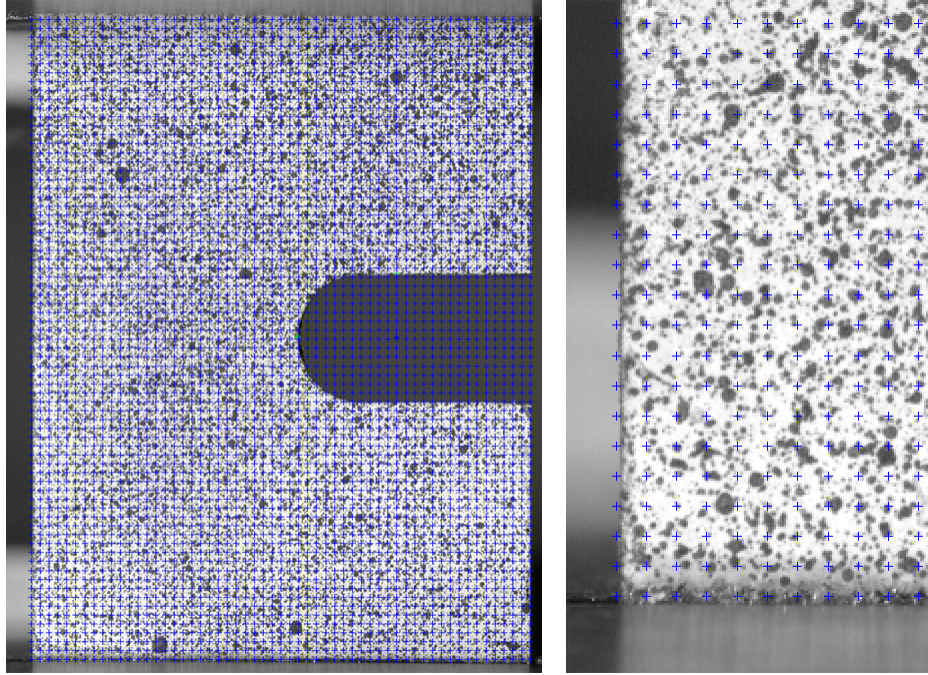


Figure 23: A control point grid designed for the calculation of strain

8-bit numerical brightness value from 0 to 255. Control point processing is done using the MATLAB function `Automate_image.m`, which starts by loading the control point grid and the list of files to be processed. It then takes the first image and the second image in the series and designates them as the base image and the sub image. The function then feeds the control points one by one into either the function `cpcorr.m` to use the correlation coefficient peak fitting sub-pixel displacement method, or `cpcorr_gradient.m` to use the gradient based method. The returned displacement values are added to `validx.dat` and `validy.dat` as they are calculated. Once all the control points on a given set of images are processed the next set is analyzed the same way until all images have been processed.

If the correlation-coefficient curve-fitting technique is used the MATLAB function `cpcorr.m`, included in the image processing toolbox, is used to calculate the normalized

cross correlation values for the given sub and base images. This function uses an algorithm that calculates the cross correlation coefficients in either the spatial or frequency domain depending on estimates of which would be faster. This fast normalized cross correlation technique was developed by Industrial Light and Magic for feature tracking in film production and was originally used in the movie *Forest Gump* in 1994 where it reduced processing time by 10-15 times [29]. For the speckle pattern and correlation size used in this thesis it was found that the spatial domain was most often used. The size of the sub and base images can be designated by editing this function file, but for this thesis it was found that there was little change in the results for a sub-image size of 10 pixels square or larger. The size chosen effects the size of the search area as well and for a couple of tests that had displacements between steps of greater then 10 pixels the sub image size was increased to 20.

Once the pixel level correlation maximum has been found the function `findpeak.m` uses the technique previously discussed in section 3 to fit a quadratic surface to the correlation peak and outputs a sub-pixel displacement value.

If the gradient method is chosen, the function `cpcorr_gradient.m` uses the gradient based method discussed in section 3 on page 26 to calculate sub-pixel displacements. Before the images are processed they are convolved with a two dimensional binomial low pass filter of order two as shown below.

$$\frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

This filter reduces the high-frequency of the images, which has been shown by [41] to reduce error when using correlation techniques that involve image interpolation. It will also be shown to reduce error in the gradient technique used in this thesis in a later chapter.

The gradient method was found to be more sensitive to sub-image size than the correlation-coefficient curve-fitting method. This is because there are more calculations that have to be performed on the entire sub and base images in this technique. Using a sub-image size that is larger than 40×40 pixels square was found to actually reduce the accuracy of the technique, presumably because the assumption of rigid body translation was no longer acceptable. A 20×20 pixel sub-image size was determined through trial and error to provide the best results and was used for this thesis.

After calculating displacements, the results are saved as two separate files: `validx.dat` and `validy.dat`. These files respectively contain the x and y control-point pixel positions relative to origin of the un-displaced first image. One item worth noting about these files is that in image processing MATLAB makes the upper left corner of the image the origin with positive y being down and positive x to the right. MATLAB uses the bottom right as the origin when dealing with all other kinds of data.

Boundary Displacements

After all the displacements are calculated they need to be processed into some kind of useful form appropriate for the desired application. For this thesis they were processed either into boundary displacements by `JP_boundary_displacements.m` or into strain data by `JP_inplanestrains.m`.

Boundary displacement data is output as the average x , y and rotation values at either end of the sample. For each image a linear curve is fit to the row of control points. For successive steps the displacement of the center of this curve and its rotation is recorded. The linear curves as well as center-point displacements of a tension and shear test can be seen in Figure 24 on the following page. Displacements

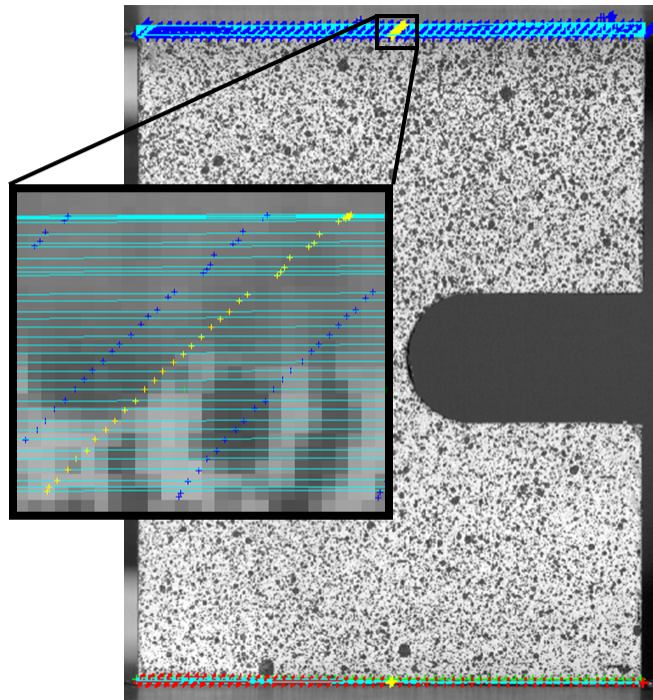


Figure 24: Boundary displacements of a test with shear and tension showing output detail. The yellow marks are the center points of the blue linear curves and the blue marks are the control points as they were tracked.

are converted to the absolute length unit used to measure the width of the sample that the user inputs when the JP_boundary_displacements.m function is called. The sample gage length is also calculated using the same conversion factor. The output of this processing is a comma-delimited spreadsheet with six columns that have the x , y , and R displacements of the linear curve fits for the two boundaries. Generally the bottom boundary displacements are subtracted from the top ones and those data are used as results. The gage length is also appended into this file as is the notch center location for IPL coupon tests.

Strain

For this project full-field strain was also a desired output from the DIC data. To meet this end, a program for calculating the strain field was written in MATLAB. It is based on the infinitesimal definition of strain shown below in Equation 17.

$$\epsilon_{ij} = \frac{1}{2} \left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right] \quad (17)$$

It is assumed that the strain on the surface of the IPL coupons is plain strain and thus the three components of strain can be written as follows in Equation 18:

$$\begin{aligned} \epsilon_{xx} &= \frac{\partial u}{\partial x} \\ \epsilon_{yy} &= \frac{\partial v}{\partial y} \\ \epsilon_{xy} &= \frac{1}{2} \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right] \end{aligned} \quad (18)$$

Before calculating strains the files validx.dat and validy.dat are rearranged into two three-dimensional arrays as shown in figure 25 on the next page with the displacement values in either the x or y dimensions for each load step stored in stacked matrices with the same dimensions as the control point grids. The original pixel locations

of each point are then subtracted so that the displacement values in the array are relative to the points original location rather than the origin of the zero-displacement image as before.

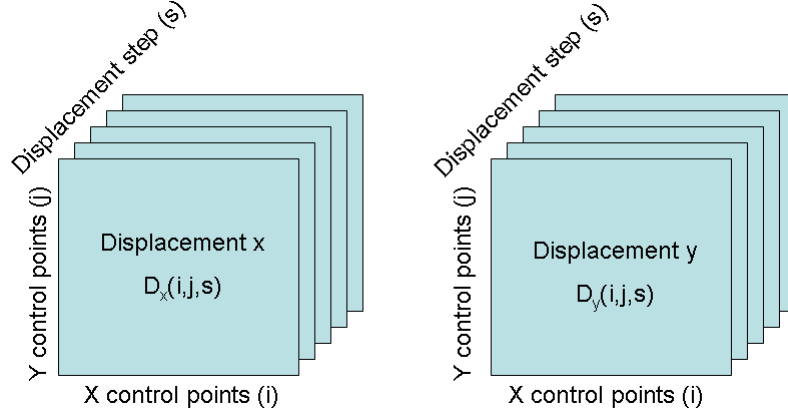


Figure 25: 3-D displacement array arrangement

After rearranging the displacements into a new format the partial derivatives were approximated using the same four-point central difference formulation of order $O(h)^4$ that is used in the gradient-based sub-pixel displacement step. In this case the spacing h_x and h_y are equal to the spacing of the control points in the x and y directions. The formulations for ϵ_{xx} and ϵ_{yy} are shown below in Equation 19.

$$\begin{aligned}\epsilon_{xx} &= \frac{D_x(i-2, j) - 8D_x(i-1, j) + 8D_x(i+1, j) - D_x(i+2, j)}{12h_x}, \\ \epsilon_{yy} &= \frac{D_y(i, j-2) - 8D_y(i, j-1) + 8D_y(i, j+1) - D_y(i, j+2)}{12h_y},\end{aligned}\quad (19)$$

where $D_x(i, j)$ and $D_y(i, j)$ are matrices of the x and y displacements for the current displacement step. For the case of the x partial derivative the D_x matrix is convolved with the mask $(1/12)(1/h)[1 \ -8 \ 0 \ 8 \ -1]$. For the y partial derivative the $D_y(i, j)$ matrix is convolved with the transpose of the previous mask. The resulting matrix contains approximations of the partial derivatives for all of the control points except for the two on the outside edges where the mask does not have full overlap.

The shear strain is calculated using the same central difference formulation and a slightly different technique. Consider the infinitesimal strain element centered at control point (i, j) shown in figure 26.

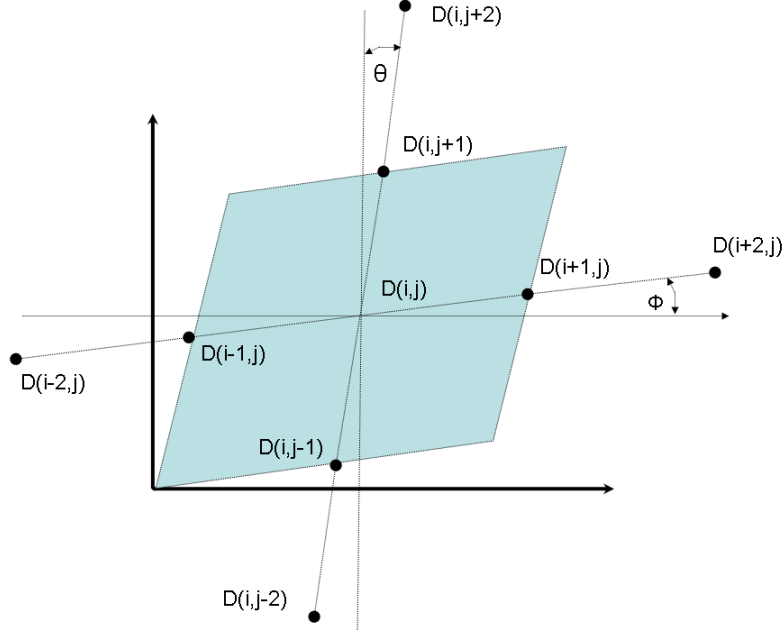


Figure 26: Shear strain element showing the points used for the central difference approximation of shear strain

In this case the items of interest are the two angles Φ and Θ . The strain is actually the tangent of these angles but they are very small and the small angle approximation allows them to be used directly. These angles can be calculated using the same central difference approximation used before, only swapping the D_x and D_y displacement matrices to swap the denominators in the partial derivatives.

$$\Phi = \gamma_{xy} = \frac{\partial u}{\partial y} \quad (20)$$

$$\Theta = \gamma_{yx} = \frac{\partial v}{\partial x} \quad (21)$$

$$\frac{\partial u}{\partial y} = \frac{D_x(i, j-2) - 8D_x(i, j-1) + 8D_x(i, j+1) - D_x(i, j+2)}{12h_x} \quad (22)$$

$$\frac{\partial v}{\partial x} = \frac{D_y(i-2, j) - 8D_y(i-1, j) + 8D_y(i+1, j) - D_y(i+2, j)}{12h_y} \quad (23)$$

(24)

Finally, the average of γ_{xy} and γ_{yx} is taken to give ϵ_{xy} as per the definition given earlier and below.

$$\epsilon_{xy} = \frac{1}{2} \left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right] \quad (25)$$

These strain data are then stored in arrays much like the displacement array in Figure 25 on page 54. They can be plotted over the images to show strain on the surface of the sample using JP_strain_visualizer.m as seen in Figure 27. This

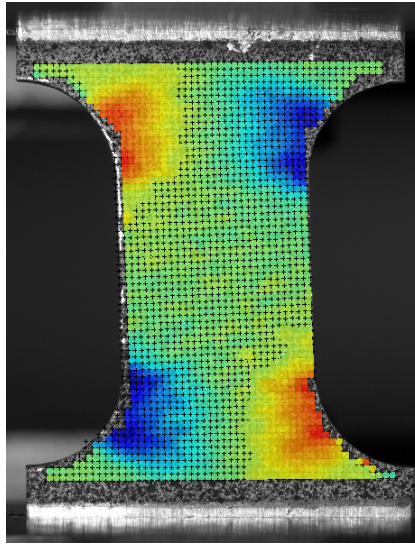


Figure 27: A plot of ϵ_{yy} on the surface of an aluminum dog bone.

figure shows ϵ_{yy} on an aluminum dog bone subjected to shear. These data can also be exported as a .dat file for further use in MATLAB for the DED method (which will be discussed further in Chapter 5 on page 93), or exported in part as a comma delimited spreadsheet file for importation into another program such as Excel.

TESTING THE ACCURACY AND PRECISION OF THE SYSTEM

The DIC system was tested in three major stages, starting at the software level. The software's ability to pick up translation and strain was tested using images that were artificially strained and translated to eliminate any error caused by image capture and processing. After the performance of the system on the software level was verified, the hardware system was isolated by looking at the displacements and strains that were manifest from images of a still, unstrained and untranslated sample. After looking at both of these situations the DIC system was used to measure strains of a simple uni-axial aluminum test and the resulting modulus and yield strength was compared to published values. Finally, an aluminum dog bone was subjected to a shear load and the resulting strains were calculated and compared to a finite-element simulation.

Artificially Generated Translation and Strain

The initial testing of the image correlation system was done by artificially creating strained and translated images in MATLAB and running the displacement and strain programs on these. Most of the image translations were done using affine transformations in MATLAB with the exception of the displacements, which were done using a Fourier filter so that the phase and amplitude of the image signal was maintained. Displacements were investigated first.

Artificially Displaced Images

To investigate the accuracy of the correlation techniques at obtaining sub-pixel displacement values a series of 61 images were created with each one displaced from

the previous by 0.05 pixels for a total displacement of 3 pixels. The original image used was the initial image from a carbon fiber test done on the IPL. The translation was performed by using the Fourier filter $\exp(-i\pi\Delta x)$ where Δx was the displacement value. This was done to remove interpolation bias and to maintain the the phase and amplitude of the image signal as recommended by [41]. Translation error was also investigated using MATLAB's piecewise linear interpolation technique and a small decrease in accuracy was noted when compared to the Fourier technique. The translation was done over a three pixel range to investigate whether or not there was any amount of error that would accumulate past the integer translations, which should have perfect correlation. In all cases tested, error was found to be small at less then 0.06 pixel in magnitude and to oscillate in a somewhat sinusoidal pattern with zeros at 0.5 and 1 pixels increments. The pixel error values for translation correlation for the gradient based method and the correlation curve fitting method with and without binomial filtering are shown below in Figure 28 on the next page. The points on this plot are the average of a representative sample of the data from the four techniques. It was found that in translation, the gradient based method with binomial image filtering had the least correlation error. The gradient method was also found to have the largest error when done without binomial image filtering. The results from the analysis of all of all 1792 of the control points from this test are summarized in Table 3. These data show that the maximum error for the gradient based method with binomial filtering is less than half of that of the others, while the standard deviation is lower by almost a factor of five. The lower standard deviation is because there are fewer outliers due to badly tracked control points in this method, when compared to the others. From here on out, the gradient based method was used at all times.

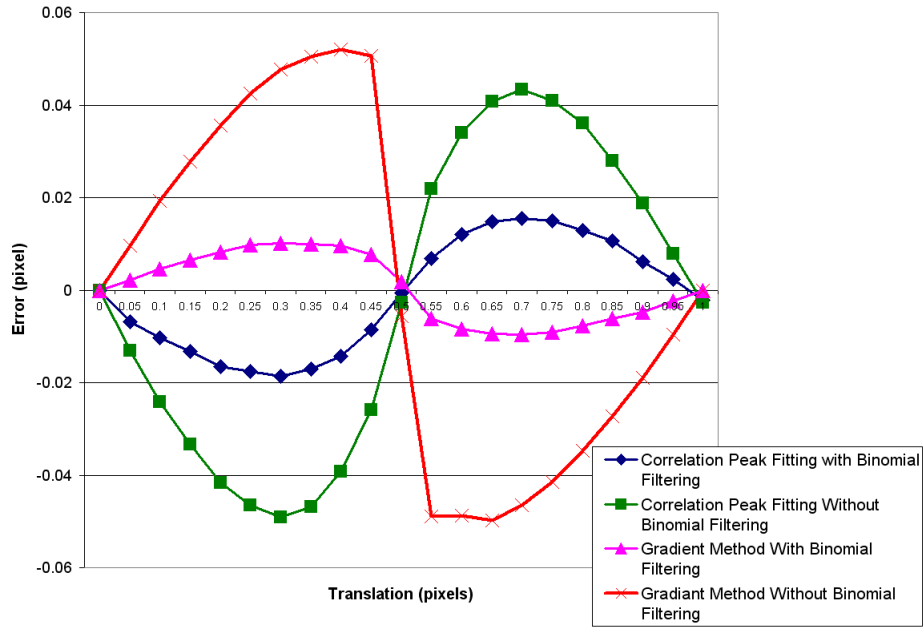


Figure 28: Correlation Error for four Different Correlation methods for a 1 Pixel Translation

Displacements were tested primarily because tracking displacements is the base-level use of DIC and it would reveal any systematic error most clearly. However, this system was primarily designed for use during composite material testing so its ability to track control points on samples under strain is most important. There were also concerns that the assumption of rigid body motion made by the gradient based technique would break down when attempted on images with large strains that needed to be alleviated. For these reasons several sets of artificially strained images were produced starting with a simple uni-axial case and the results were compared to ASTM extensometer standards.

Artificially Strained Images

The strained images in this thesis were produced by using a piecewise-linear interpolation technique. This technique is one of the most basic forms of interpolation

Table 3: Image Translation Correlation Systematic Error for a Three Pixel Displacement

	Max Error (Pixel)	Std. Dev. (Pixel)
Corr. Coeff. Peak fitting with Binomial Image Filtering	0.074	0.013
Corr. Coeff. Peak Fitting without Bin. Image Filtering	0.102	0.018
Gradient Based with Binomial Image Filtering	0.019	0.003
Gradient Based without Binomial Image Filtering	0.080	0.016

that can be used, however it is quite accurate for simple strains. An example of linear interpolation is shown for a one dimensional case in Figure 29. The images are first

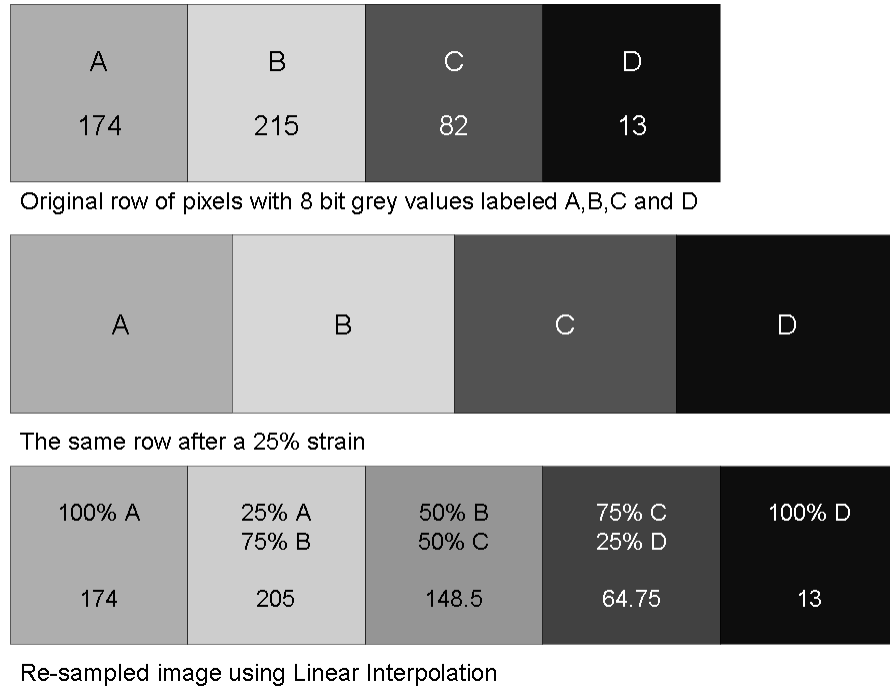


Figure 29: Linear Interpolation on a Row of Grey Values

strained continuously as shown in the second step above, then re-sampled into the discrete pixel sizes as shown in the final step. It should be noted that some of the

resulting gray values contain a decimal. These decimal values are rounded to the nearest integer, which produces a round-off error that is at its maximum when the interpolated pixel values contains a decimal of .5. This error is present in the strained images produced and can be considered a factor in the total error that is present in the strain data produced. However, the same error occurs in real testing when the image produced by the lens is discretely sampled in 8 bits so the round-off error with the artificially strained images should be quite similar to what happens during actual tests.

The unidirectional strain tests were conducted following the ASTM E 83-06 standards for Classification of an Extensometer System [5]. These standards set limits on initial fixed error and relative error for six different classifications. The limits are summed up below in Table 4. The images for the test were produced

Table 4: ASTM Classification of an Extensometer System

Classification	Fixed Strain error limit (m/m)	Relative Strain error limit (% of strain)
Class A	± 0.00002	± 0.1
Class B-1	± 0.0001	± 0.5
Class B-2	± 0.0002	± 0.5
Class C	± 0.001	± 1

by using MATLAB to artificially strain an actual IPL test image by 10% in steps of 0.25% for a total of 40 steps. The gage length of the sample was measured as 27.8mm using the width for calibration. The control point grid spacing along this gage length was correlated at two different distances, 50 pixels and 15 pixels, to give two different sets of data. The strain data was calculated by averaging the strains at all the control points in the y direction for each step of the test. An ASTM standard report for these tests is included in the appendices. The resulting error was calculated by taking the absolute value of the difference between the actual strain of the image

and the strain returned by the software. This error is shown below in Figure 30 along with the piece-wise representation of the ASTM Class B-1 and B-2 error limits for an extensometer system. Looking at the figure it is apparent that the error in the

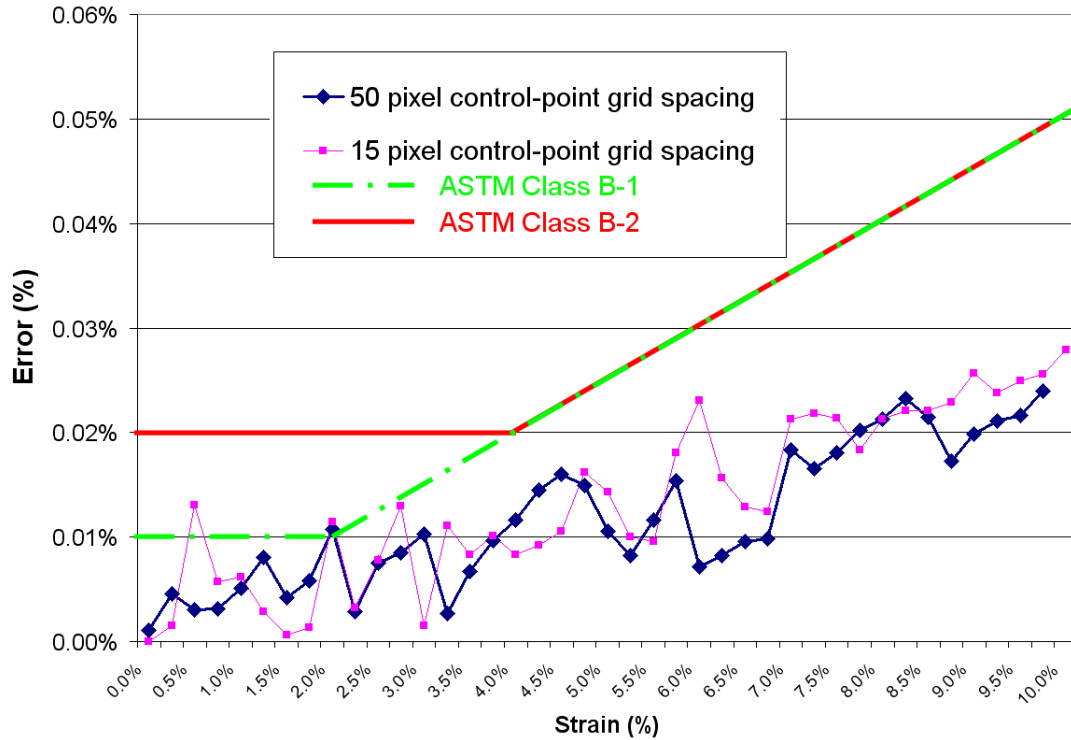


Figure 30: Strain Error and Extensometer System Class

software is well within ASTM B-2 classification and almost ASTM B-1. It is also apparent the the two different gage lengths tried have little effect on the error of the strain calculations. When looking at the strain data over the entire sample surface, some interesting observations can be made. The strain values returned by the central-difference formulation occur in a repeating pattern along the strained direction of the sample. The measured strain over the gage section at an actual strain of 0.25% is shown below in Figure 31. The averaged value of this strain is quite close to 0.25% with an error of only .4% but the error at any given point would be larger. Once the strains are large enough the pattern condenses into a sawtooth pattern with more

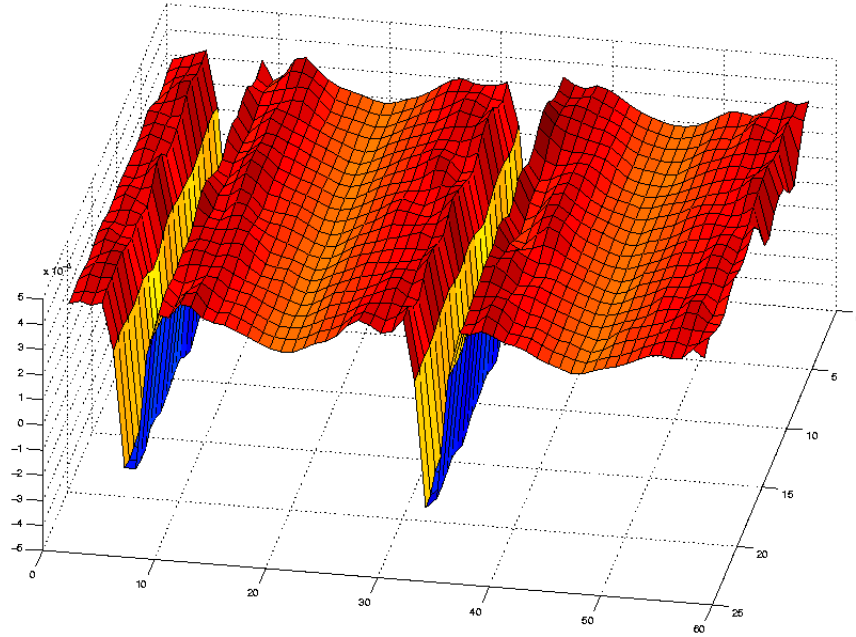


Figure 31: Measured Strain When Actual Strain = 0.25%

apparently random noise as seen below in Figure 32 on the next page. Ostensibly this variation would be due to a combination of the round-off error present in the strain calculation and the central difference formulation that is used to calculate strains and determine the sub-pixel displacements they come from. However, determining the actual reason behind this variation is beyond the scope of this thesis and will be left for future work.

The uniaxial strain test was conducted for the sake of comparing the system to established standards. Further testing was done with with shear and biaxial strains for the sake of investigating the system's usefulness for measuring complicated strain states like those on the composite coupons used in the IPL. For these tests the same IPL image from before was used to make new sets of images different combinations of strains. An example of an image from a test with a complicated strain state can

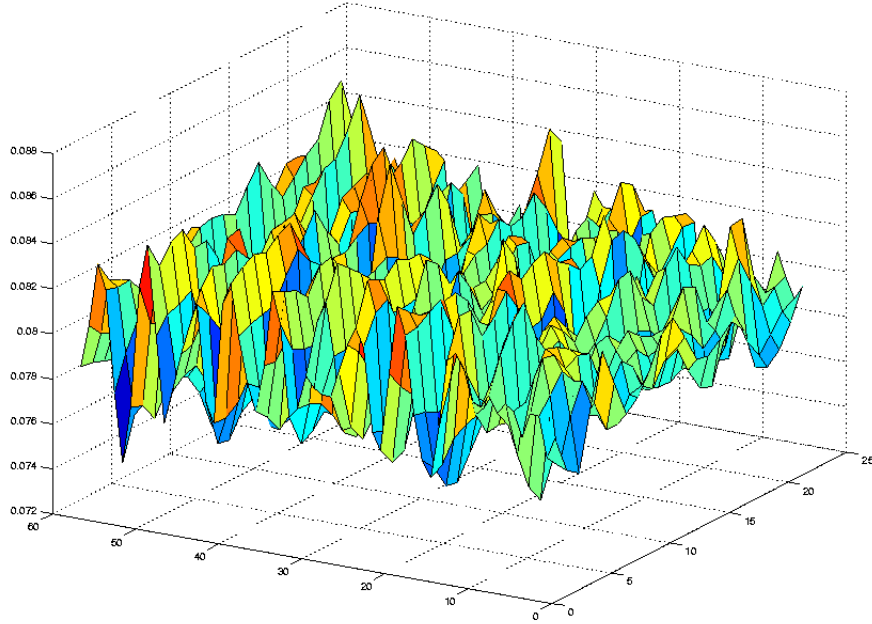


Figure 32: Measured Strain When Actual Strain = 8%

be seen below in Figure 33 on the following page.

This is a typical image produced by the MATLAB `automate_image.m` which shows the original control point locations in green and the current control point locations in red. It also shows progress and estimated time as well as the number of control points being tracked.

Three plane-strain cases were investigated, the first was a combination of ϵ_{yy} and ϵ_{xx} . The second was only shear strain ϵ_{xy} . The third and final test included all three strains ϵ_{xx} , ϵ_{yy} , and ϵ_{xy} . The first test was a combination of 10% ϵ_{yy} and 5% ϵ_{xx} . Strains were calculated from the displacements of 1545 control points and averaged. The results can be seen below in Figure 34 on the next page. The strains calculated by the software in this test have an average fixed error of 0.0087% in the y axis and 0.0125% in the x axis. When compared to the ASTM extensometer class standards

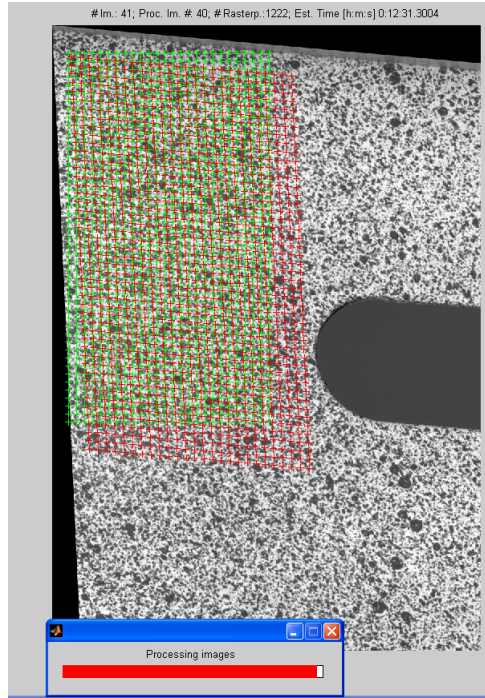


Figure 33: Artificially created ϵ_{xx} , ϵ_{yy} , and ϵ_{xy} strain

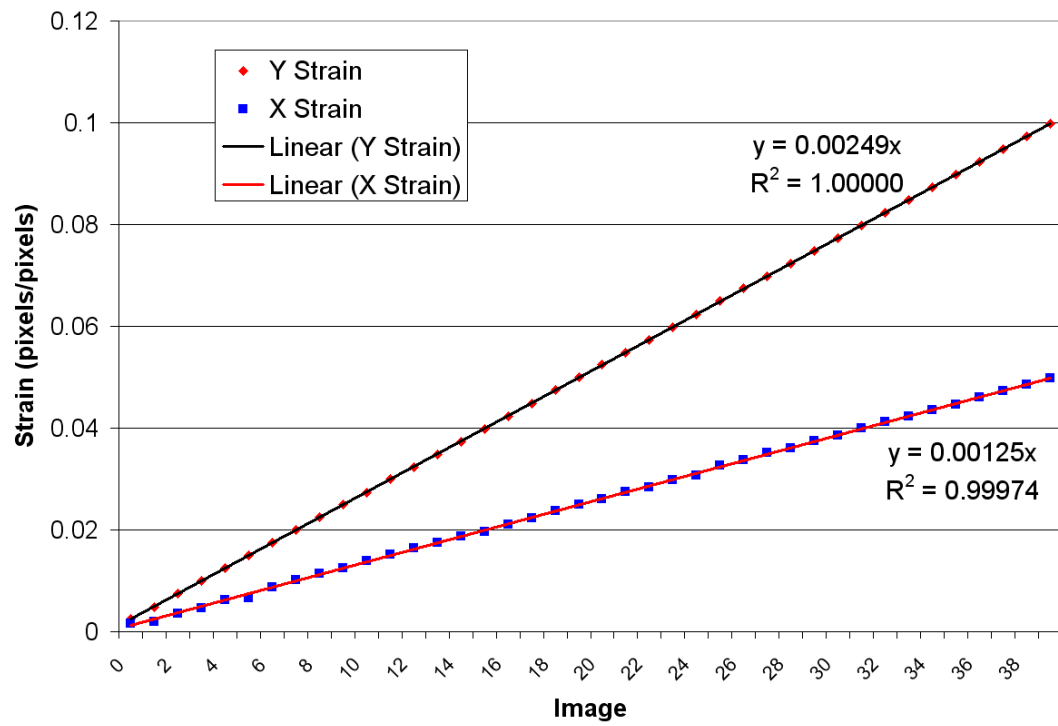


Figure 34: Measured Strain for 5% X Strain and 10% Y strain

the results for this test are just barely outside of the B-2 class. This combination of strains has larger error then the single dimension strain case, although as can be seen in Figure 34 on the preceding page the results it produces are still quite linear.

The next strain case investigated was shear strain. Using the same methods as before, 40 strained images with shear strain ϵ_{xy} increasing from 0 to 5% were generated. The measured shear strain is plotted for each image in Figure 35. The resulting average fixed error for this test was 0.0035%. This would be within the limits of ASTM B-2 extensometer classification. As a final test a combination of

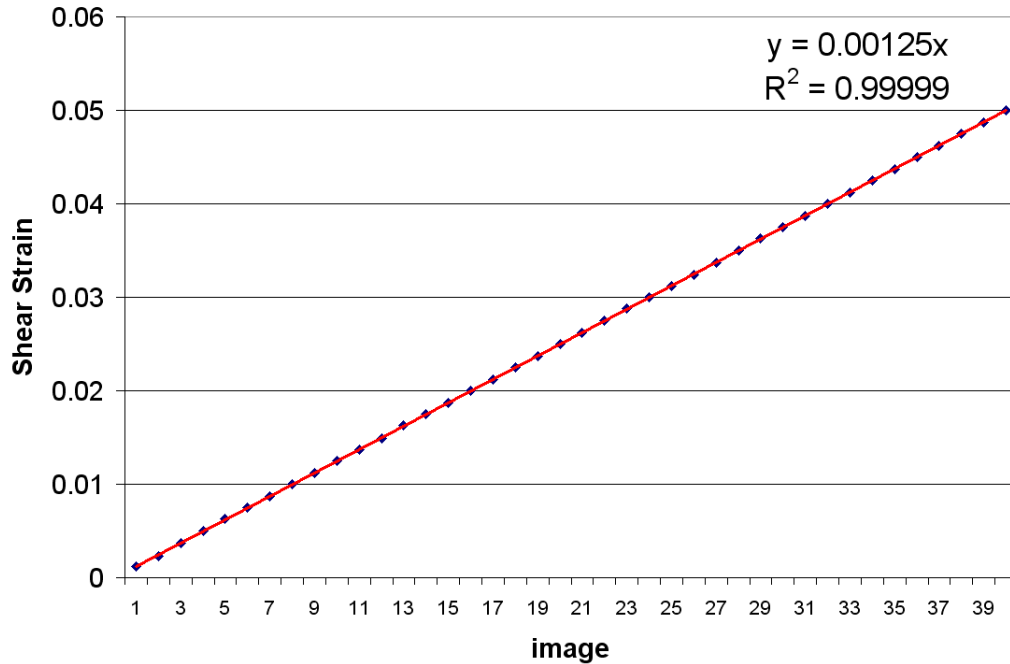


Figure 35: Measured Strain for 5% Shear Strain

strains 10.00% ϵ_{xx} , 6.67% ϵ_{yy} and 7.50% ϵ_{xy} was created and tested as before. The errors in this test were all really close to the ASTM B-1 standards with fixed error starting very low and growing slowly with each successive image. More so then either of the previous three tests the error in this test grew with each successive image, as can be seen below in Figure 36 on the following page. The reason for this is not clear,

although it may have something to do with the fact that the strain state is more complicated than in previous tests. Even with the error increasing it stayed under the limits of ASTM B-2 classification.

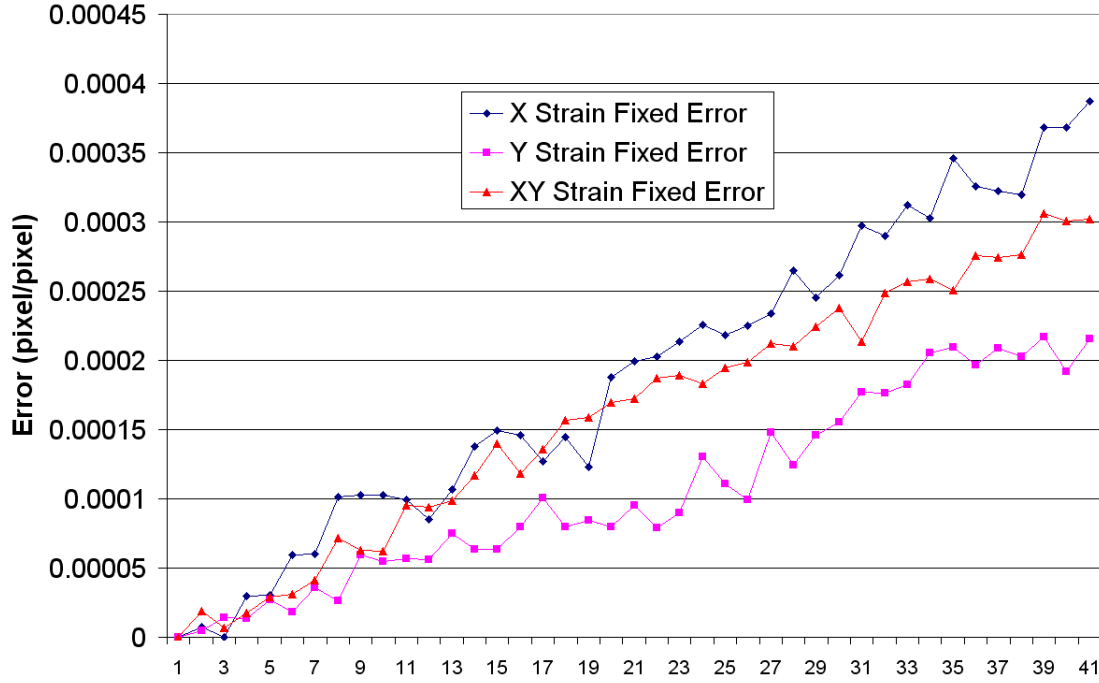


Figure 36: Fixed Error for Images With ϵ_x , ϵ_y and ϵ_{xy}

Measuring Precision with a Stationary IPL Test

The previous tests using artificially created strains were evaluated for the sake of verifying that the software works as intended. Further testing was then done using images captured during actual testing on the IPL to investigate the hardware and image processing portion of the system. The first test consisted of a sequence of 100 images of a specimen mounted in the IPL without any application of loads. One of these images can be seen in Figure 37 on the next page. The purpose of this test was essentially to investigate all of the other sources of error that are not from the

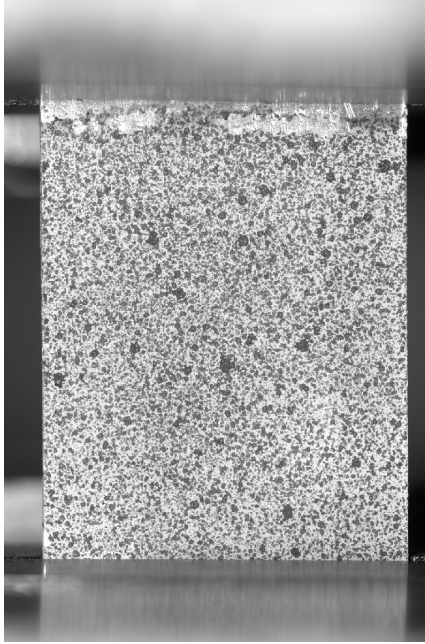


Figure 37: 1 of 100 Images of a Stationary Sample in the IPL

image correlation itself and had not been looked at yet with artificially created strain and displacement. The first thing that was investigated in this test was the average absolute displacement of all the control points on the sample for each step. The absolute displacement of the sample during a test is not really important because all of the results deal with relative displacements between control points, but they were investigated to get an idea of how much motion was picked up by the image correlation due to vibrations in the frame and mounting system. The resulting displacements varied with a standard deviation of 0.072 pixels in the y direction and 0.18 pixels in the x direction. This and all the data from this section can be seen below in Figure 37. Given the scale of the image, this translates to approximately 0.005 mm in the x direction and 0.0002 mm in the y direction. This could be caused by something as small as the vibrations produced by the camera when the shutter and mirror flip for each image. The relatively large standard deviation in the x direction is due to an overall increasing trend in the data, the cause of which is unknown. One possibility

is that it is from the mirror of the camera flipping for each shot and causing a very slight rotation over the course of many exposures. The variation in displacement over

Table 5: Stationary IPL Sample DIC Standard Deviations

	Standard Deviation
X Displacement	10.1×10^{-04} pixels
Y Displacement	8.49×10^{-04} pixels
X Strain, ϵ_{xx}	3.4283×10^{-05} pixels/pixels
Y Strain, ϵ_{yy}	3.3004×10^{-05} pixels/pixels
Shear Strain, ϵ_{xy}	2.2489×10^{-05} pixels/pixels

the surface of the sample in each step was investigated next. Since any motion that was picked up during this test would have to be purely translational, any variation in displacements across the surface of the sample can be linked to systematic error. To look at these errors, the standard deviation of the displacement values for all of the control points at each step were calculated and then averaged for the whole test. The average standard deviation in the x direction was found to be 0.00101 pixels while the average standard deviation in the y direction was 0.00085 pixels, as seen in Table 5. This is actually lower than the standard deviation of the displacement of the artificially translated images by a factor of 3. This is not all that surprising because the displacements that were picked up in this test were almost all under 0.25 pixels and the maximum standard deviation in displacements occurs at 0.5 pixel displacement. This is quite reassuring however, as it ostensibly indicates that the systematic error being introduced by the hardware system and image processing is small enough to be negligible. Finally, strains were calculated off of all the images in the series to see what kind of effect the error in the displacements would have. As would be expected, the strains were very low with an average ϵ_{xx} of -4.64753×10^{-6} and ϵ_{yy} of 1.62684×10^{-5} . The standard deviation for ϵ_{xx} was the largest at 3.4283×10^{-05} as seen

in Table 5. These values are very low and indicate that the error being introduced into the displacement data is not affecting the strain calculation negatively.

Uniaxial Tension on 6061-T651 Aluminum

After analyzing the behavior of the DIC system with artificial strains and displacements and a static sequence of images testing with aluminum was done to look at its performance in actual testing. Aluminum was used because it is an isotropic material whose properties are well known and easily analyzed. The first two tests were on narrow strips of 6061-T6 aluminum. They were tested in uni-axial tension on the IPL, an image from one of the tests is shown in Figure 38. For both tests the

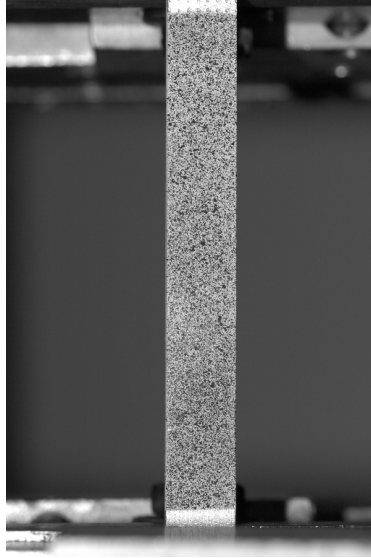


Figure 38: An Aluminum Tension Test Sample

cross sectional area was 35.2 mm^2 and the gage length was 100mm. The first test was done with a maximum displacement of 0.762 mm in order to get data after yield. The second test was done to 0.508 mm in order to get more detailed data in the linear elastic region. For these tests the strains were measured three different ways:

- Using the MATLAB DIC system developed for this thesis
- Using LVDT data from the IPL
- Using a trial version of the commercial DIC package VIC-2D [44]

In the MATLAB system a control point spacing of 15 pixels was used and a control point grid was constructed over the entire sample. Strain was calculated by averaging the strain over the surface for each step. The results from the first test can be seen below in Figure 39. This figure shows the yield response of the aluminum as shown by

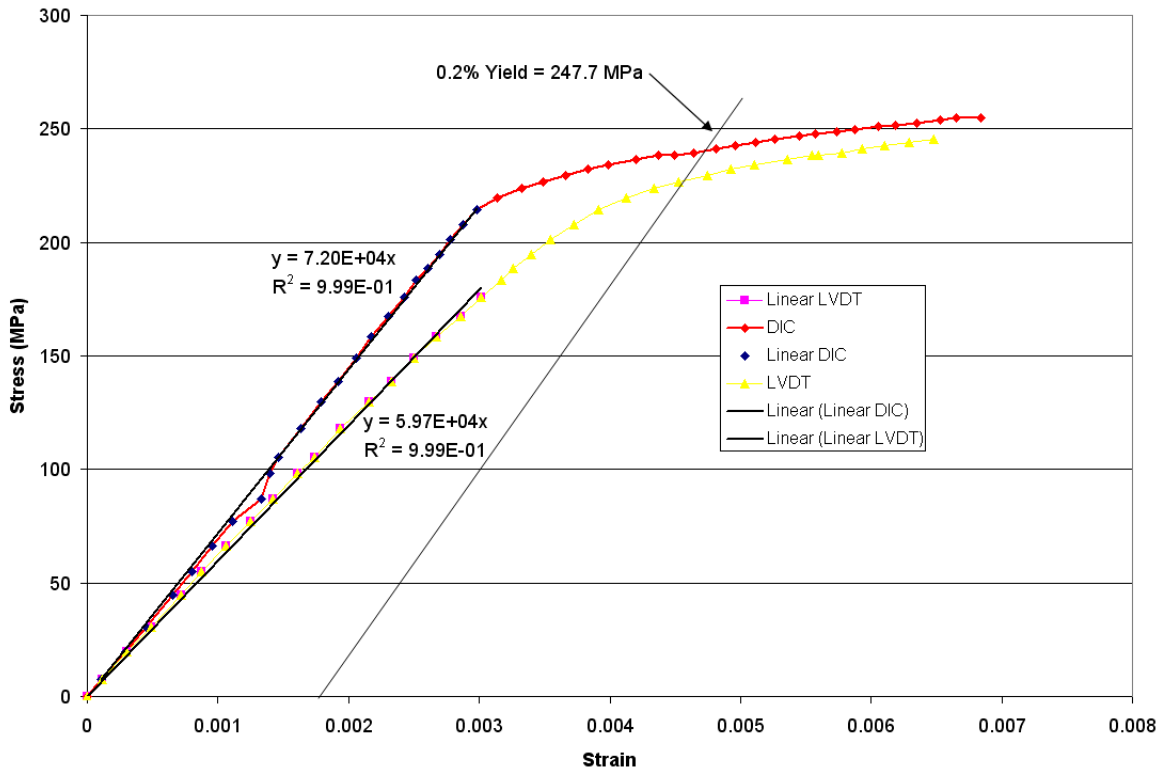


Figure 39: Modulus and 0.2% Offset Yield Strength of 6061-T651 Aluminum

strain calculated through the DIC technique and stress calculated from loads captured by the IPL load acquisition system. The Young's modulus (E) can be seen as well as the R^2 value of the linear curve fit to the linear portion of the data. The 0.2% offset

yield strength (σ_y) was calculated as well and is on the figure. The induced ϵ_{xx} strain was also measured and Poisson's ratio was calculated. These results are summarized below in Table 6 on page 74. The Poisson's ratio (ν) was calculated from the last half of the strain values because the ϵ_{xx} values for the first half of the test were below the precision of the strain calculation program as had been determined earlier and were not very consistent. The LVDT data can be seen on this figure as well, and illustrates a few of its shortcomings. The modulus reported by the LVDT's is approximately 60 GPa, which is significantly lower than the stiffness reported by the DIC system and the published stiffness. The response it shows is not very linear, making a gradual curve of decreasing stiffness without a visible yield point. The reason for this strange behavior is that the LVDTs are mounted on the grips. As a consequence, there is a complex interaction between the grips and the sample that transfer the force to the gauge section of the sample by a shear force applied to the gripping area of the coupon. There is a combination of shear surface displacement and slippage of the grips that takes place and needs to be accounted for before using LVDT data. This relationship has been modeled with some success by Dongfang Huang in [27] using a three dimensional finite element model. However, the use of the DIC system negates the need for this complex extra step by taking displacements directly off of the sample itself. This is especially useful during tests involving shear and rotation where gripping effects are more apparent, of which there is more discussion in [40]. After calculating these material properties the results from VIC-2D were then compared to the MATLAB results as can be seen in Figure 40 on the following page. In this figure there is significantly more visible variation in the VIC-2D curve when compared to the MATLAB DIC curve. The linear curve fit to the linear portion is also shown with the coefficient of determination R^2 value for both curves. VIC-2D uses an iterative technique for measuring displacements that has been shown by [35] to have less error

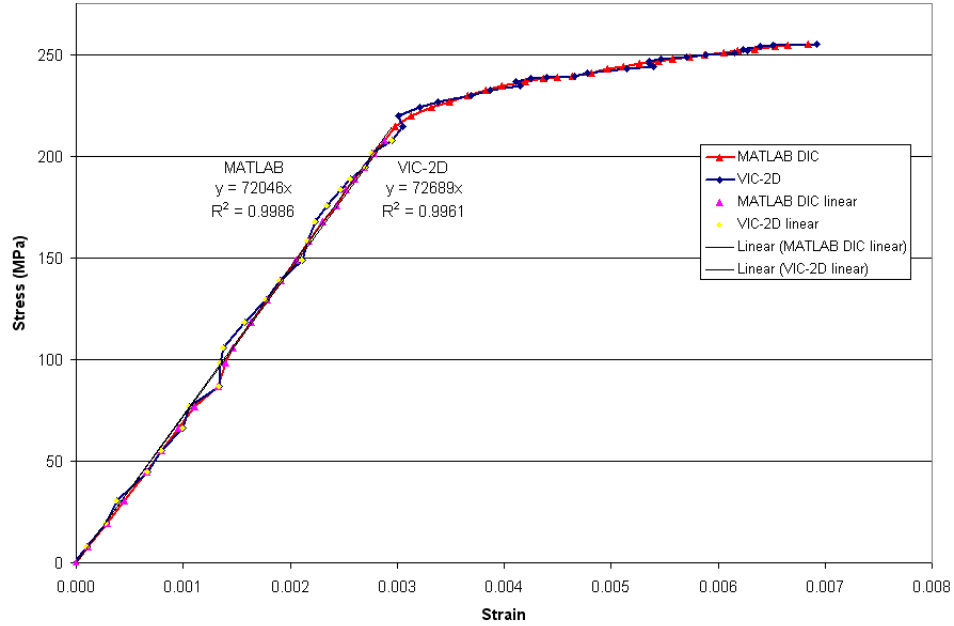


Figure 40: Stress vs. Strain of 6061-T651 Aluminum as Measured by VIC-2D and the MATLAB DIC Program

than the gradient based technique used for this thesis so its displacement calculation should be more accurate. The control point spacing used is 5 pixels by default, three times smaller than that used by the MATLAB program. VIC-2D calculates strains by fitting a polynomial to the displacement data of the 15 control points surrounding the point in question and then measuring the slope of the line. The central difference used to calculate strains in this thesis only uses 4 control points but since the control point spacing is three times larger the width of the area used is only 15 pixels smaller than that of VIC-2D at 60 pixels as opposed to 75. The reason why the VIC-2D results appear to be worse than the MATLAB program is unknown, but it likely involves some of the previously mentioned differences in approach. The results from the second test were not nearly as consistent as the first test. This was because the LVDT feedback system on the IPL was turned off and the actuators themselves were used to define each displacement step. This is not a good way to get evenly spaced

Table 6: 6061 T6 Aluminum Properties from Two Tests

Property	DIC 1	DIC 2	Pub. Values	% Error Test 1	% Error Test 2
E	70.8 GPa	66.5 GPa	68.9 GPa	-2.84%	3.50%
σ_y	247.78MPa	N/A	276 MPa	10.3%	N/A
ν	0.30	0.38	0.33	7.22%	-16.43%

data points as can be seen in Figure 41. This data was still included because the

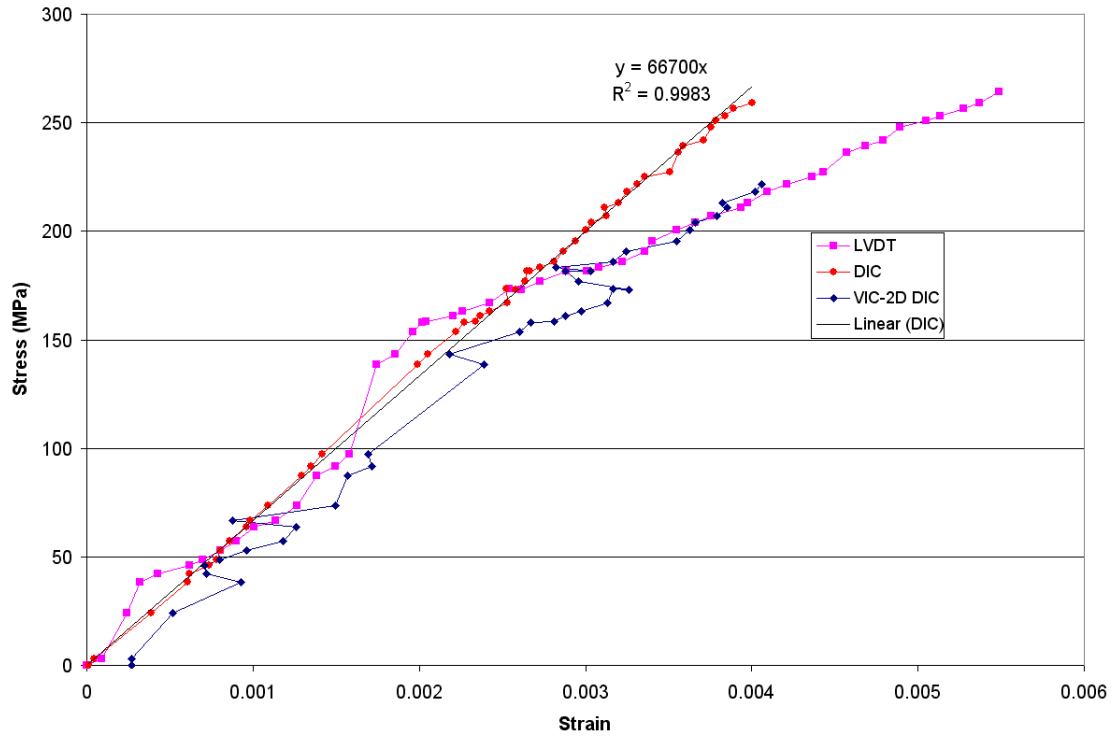


Figure 41: Modulus and 0.2% Offset Yield Strength of 6061-T651 Aluminum

MATLAB DIC system still produces a linear stress/strain curve, much more so than the strain from the LVDT's or the strains from the commercial DIC system VIC-2D. Once again VIC-2D does not appear to produce very good results for reasons unknown. The Young's modulus and Poisson's ratio estimates from this test are also displayed in Table 6. The percent error in this test was almost totally opposite of that of the first test but again showed the best results for Young's Modulus.

As can be seen from the results of both tests, the lowest percent error was when calculating the modulus. The percent error for the yield strength and Poisson's ratio is significantly higher than that for the modulus. There are several sources of error for these tests that could be responsible. The most major source of error is likely from the load acquisition system of the IPL. As has been shown by Aaron Collett in [17]b the y load error can be greater than 10% in the y and x directions. This error drops to less than 5% as the loads increase to the maximum loads of these tests however. Another source of error is in measuring the samples. The samples were designed with a very small cross sectional area to ensure that yielding them would be well within the capabilities of the IPL. Because of the small cross section minute variations in the width or height have a large effect on the stress calculation and while they were measured very carefully at multiple locations with a caliper, error would still have been possible. Out of plane deflection is another source of error mentioned by Collett that most likely produced some error in this test.

Shear Test on a 6061-T651 Aluminum Dog bone

The final test conducted with the MSU IPL was a pure shear test done on an aluminum dog bone. For this test a Finite Element Model (FEM) was made using ANSYS software and the strains produced by the model were compared to those measured with the MATLAB DIC software as well as those from VIC-2D. For these tests a geometry was created that would provide a large surface area for the DIC technique and provide a large range of strains to compare. The chosen geometry can be seen in Figure 42 on the next page. This sample was tested to a shear displacement of 1.27 mm in the x direction according to the IPL coordinate system. The test was done as before with 50 steps and LVDT position correction. In all shear

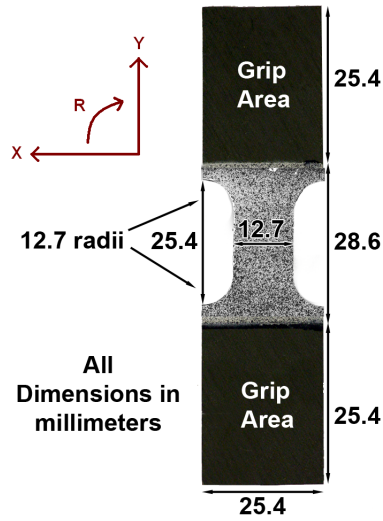


Figure 42: 6061-T651 Aluminum dog bone Geometry for use with the IPL

testing that has been done with the IPL, unwanted rotation and y displacement occurs, and this test was no exception. As was done in the most recent testing for the dissipated energy density function, boundary displacements were calculated using the JP_boundary_displacements.m MATLAB function and the unintended y and R displacements were included in the boundary conditions for the FEM. These boundary displacements can be seen below in Figure 43. For the most part the displacements

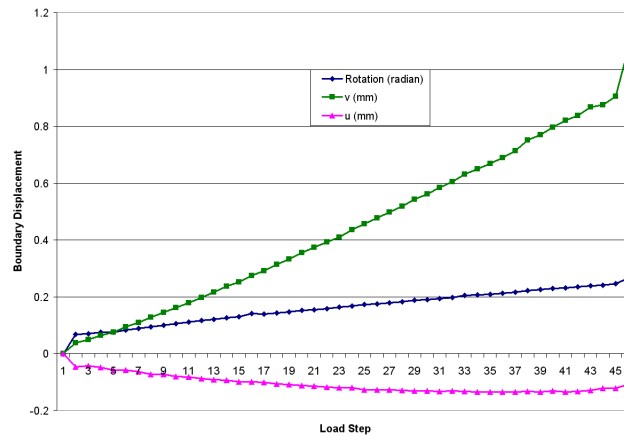


Figure 43: Boundary Displacements for a Shear Test on a 6061-T6 Aluminum Dog bone as reported by the DIC system

are quite consistent, although there is some interesting behavior at the beginning and end of the test. The jump in all displacements at the beginning comes from the difference in the readings of the LVDT position system used for controlling the IPL and the DIC boundary displacement measured directly off the sample. On the first step the LVDT system under-reported the displacement, ostensibly because of some form of mechanical slack in the system that had to be taken up. The cause of the jump in the end of that data however is unknown. Another detail worth noting is the the final displacement measured from the sample was just over 1mm, which is almost 16% less than that reported by the LVDT system. This large difference is not uncommon between the two displacement measurement methods during shear testing and will be discussed in further detail later when talking about the use of the DIC program in testing for the DED function.

After determining the boundary displacements they were inserted into the FE model and strains were compared to the DIC strain results. The dog bone was modeled in ANSYS using 8-node quadrilateral Plane-82 elements, which were chosen because they are a 2-D element that work well for meshes that have curved edges like the dog bone. The mesh was refined until the solution converged. A contour map of the ANSYS element solution for the ϵ_{yy} strain at displacement step 49 of the test can be seen below in Figure 44 on the next page. The DIC counterpart of this image is a labeled contour map of the strain matrix and is right next to the ANSYS solution as Figure 45 on the following page.

The strains reported from the MATLAB DIC results can also be compared to those reported by VIC-2D. Those results are plotted below side by side as Figures 46 on the next page and 47 on the following page. Several points on this these figures are picked out for the sake of comparison.

A qualitative assessment of similarity can be made by looking at the contours in

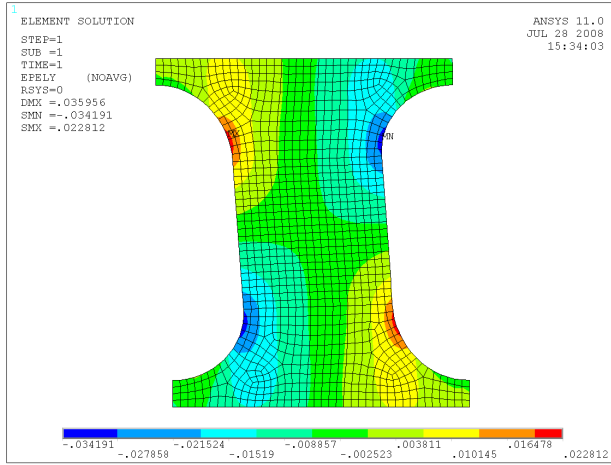


Figure 44: ϵ_y strain result from the ANSYS model. This figure also shows the element density.

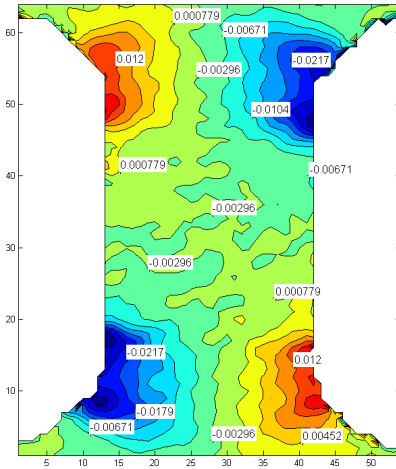


Figure 45: ϵ_y strain as calculated from the MATLAB DIC results.

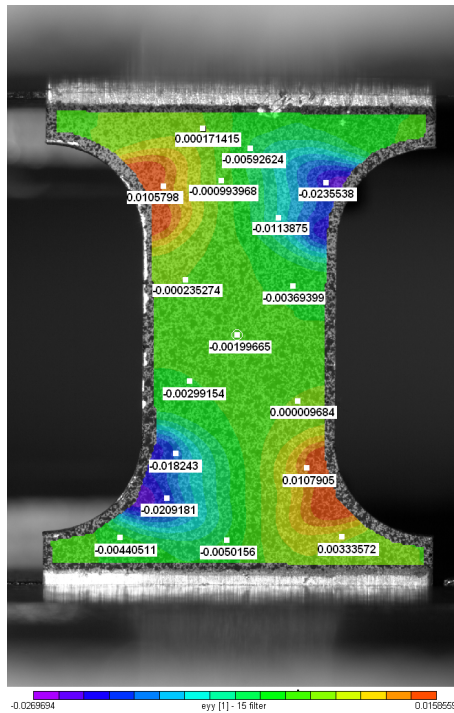


Figure 46: ϵ_y strain result from VIC-2D

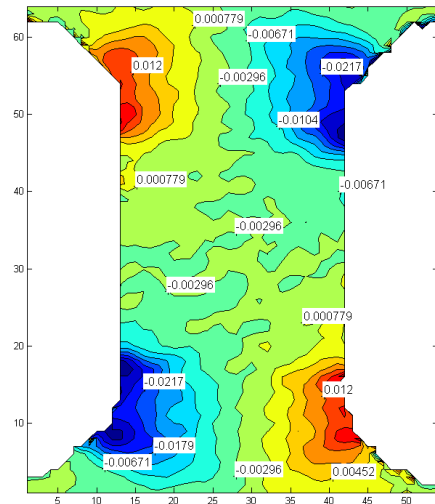


Figure 47: ϵ_y strain as calculated from the MATLAB DIC results.

the strain plots. A more quantitative result was obtained by made by looking at the strains in specific locations on the contour plots, however this is still hard to quantify as they don't match up perfectly. The minimum and maximum values are something that can be compared quantitatively however. These values from the FEM solution and the DIC results from both the MATLAB DIC system and VIC-2D as well as their percent differences are summarized in the following Table 7.

Table 7: DIC Maximum and Minimum compared to FEM Results

	FEM	MATLAB DIC	VIC-2D DIC	Matlab Dif	% VIC-2D Dif
Max ϵ_{yy}	0.0228	0.0255	0.0158	-11.84%	30.7%
Min ϵ_{yy}	-0.0342	-0.0315	-0.0269	7.89%	13.4%

As was the case with the modulus tests, the MATLAB DIC system appears to produce better results. In this case it is not a fair comparison however as VIC-2D does not report strains as close to the edges of the dog bone as the MATLAB system does for ϵ_{yy} due to the fact that it automatically cuts out the outside 7 control points in the x axis. The MATLAB solution was manually trimmed to remove any central differences that don't have full overlap in the y direction on the sample surface but was trimmed all the way up up the the edge of the sample in the x direction when possible so it includes more points. When looking at the MATLAB and VIC-2D solutions side by side it can be seen that the MATLAB solution's results are not as smooth as those produced by VIC-2D, but the strain does appear to match up well.

Summary of Testing Results

The software side of the DIC system developed for this thesis was tested first. Displacement accuracy was found to be within 0.02 pixels on artificially displaced

images during pure translation when using the gradient based method. Strain was looked at in multiple combinations from uniaxial ϵ_{xx} to a combination of ϵ_{xx} , ϵ_{yy} and ϵ_{xy} . With the exception of a few points the results were within ASTM class B-2 standards, meaning that the strain error is less then 0.0002 m/m or $\pm 0.5\%$ of the strain, whichever is greater.

The precision of the hardware system was investigated using a series of 100 images of a stationary sample in the IPL. In this test the standard deviation of the displacement values for all the control points at each step was taken, then averaged for all 100 steps. The maximum standard deviation for the displacements was in the x direction and was 0.00101 pixels. In the y direction it was less at 0.00085 pixels. The standard deviation for the strain at each step was also calculated, this was found to be highest for ϵ_{xx} at 3.4283×10^{-05} m/m. These results can be found in Table 5 on page 69.

The DIC system and the IPL were then used to run a set of two uniaxial tests on 6061 aluminum pieces. The first test was run past the yield point of the sample and second test was run in more detail within the linear portion of the samples response. Young's modulus was calculated for both tests. In the first test it was calculated as 70.8 GPa for a percent error of -2.84% and in the second test it was calculated as 66.5 GPa with a percent error of 3.5%. The yield strength was also measured on the first test with a percent error of 10.3%. Finally, Poisson's ratio was calculated for both tests using ϵ_{xx} strain data from the sample in addition to the ϵ_{yy} data used before. For the first test it was measured as 0.3 for a percent error of 7.22% and for the second test it was measured as 0.38 for a percent error of -16.43%. These values are all summarized in Table 6 on page 74.

Finally an aluminum dog bone was tested in shear in the IPL and modeled in ANSYS as a finite element model so that strain results could be compared between

the two. The maximum and minimum ϵ_{yy} strains were compared to the ANSYS model and found to differ by 11.84% for the maximum strain and 7.89% for the minimum strain. It should be noted however that these are very localized peaks and that strain values across the surface of the sample also appear to correlate well.

APPLICATIONS IN COMPOSITE MATERIALS TESTING

Most of the testing done with the DIC system was on coupons with the IPL coupon geometry shown previously in Figure 8 on page 14. The first testing used fiberglass/epoxy coupons and a different camera and lens to track boundary displacements at only a few points during the tests. The second set of tests was done with carbon/epoxy coupons and the camera and lens combination previously outlined to track boundary displacements at all points during the tests and to measure full field strain.

Boundary Displacements on Fiberglass/Epoxy Coupons

The first application of the DIC technique was calculating boundary displacements with James Schmitt in 2007 for a project that was attempting to fit a FE model with a bi-linear constitutive response to composite material test data. The testing for this project was done on fiberglass/epoxy coupons with the standard IPL geometry. The epoxy resin used was Epon 826 and the laminate was made using hand lay up and two layers of CDB200 Triax fabric from Owens Corning for a layup of $[0/45/-45]_s$. These tests were done with a different camera and lens than the current system. The camera was a Canon Digital Rebel XT, which is very similar to the Digital Rebel. The main difference between the two cameras for this application is a 21% higher sensor resolution of 3456×2304 pixels. However, this only results in an increase of 15% in line width to picture height resolution as tested in [4]. Any advantage in resolution of the camera in these tests over the final configurations was likely lost by using a much less suitable lens anyway.

The lens used for this test was a Tamron 18-200mm zoom. It was used at 200mm and had noticeably poor resolution, especially compared to the Tokina macro that was later acquired for this project. The focusing distance on the lens was not close enough to be able to fill the entire field of view with the sample either, so even with the increase in resolution provided by the camera, the pixel width of the IPL coupons in the images taken was 40% less than in the current system. All of the measurement error in the DIC software is relative to the conversion factor that converts between pixel displacements to inches or millimeters, so 40% less image resolution translates into 40% more error with all other things being equal. The camera was mounted on a tripod and positioned as close to the sample as the lens was capable of focusing for these tests and images were taken manually using an infrared remote control so as not to disturb the position of the camera.

Four different tests were done, one of pure y displacement, one of pure x , one of x and y , one with an opening rotation and one with a closing rotation. All of the tests were done with 20 load steps. Two images were taken at the first step, one of which would serve as a zero displacement calibration point. After that images were taken every third step until damage began to accumulate, then they were taken every step until the test was over. This resulted in between 5 and 12 images for each test. Image processing and importation into MATLAB was done as outlined previously. Displacements were calculated using the correlation peak curve fitting technique as the script for the gradient method had not been developed yet. Boundary displacements were then produced using the script `JP_boundary_displacements.m`, which was developed for this project. Figure 48 on the next page shows the y boundary displacement points for the combination x and y test.

After these tests it became apparent exactly how much difference there was between the IPL LVDT displacement readings and the actual boundary displacements

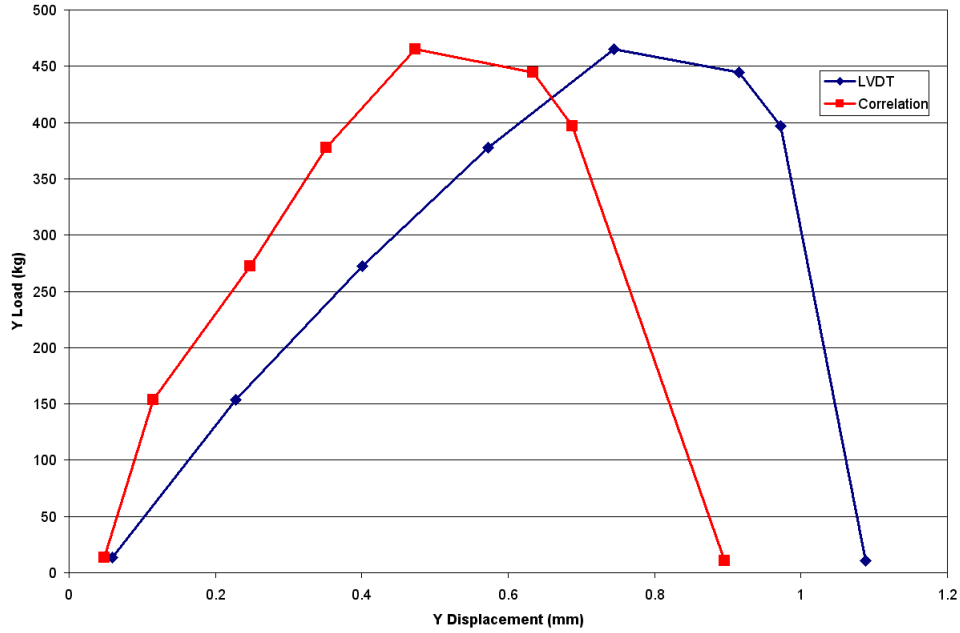


Figure 48: Boundary displacements in the y direction for a combination x and y test on a fiberglass/epoxy IPL coupon

of the sample. This difference is easy to see on the above figure as an almost 100% smaller boundary displacement. This difference was present in all of the data, but was largest in the tension test because the forces involved were larger. Part of this displacement difference is due to the slightly smaller gage length that was used but is mostly due to gripping effects. These gripping effects are mentioned in [17] and [27] and among other things include shear deformation of the coupon surface due to loads being transferred from the grips through shear.

For this test the boundary displacements were exported into a spreadsheet and then imported into ANSYS for use on a simulated IPL coupon. The first goal of the project was to determine the six material coefficients that determine the linear response of each ply based on boundary displacement data from testing. After determining the linear coefficients the model was modified to have a bi-linear response by adding six more strain values that indicate a critical and ultimate strain to define

the second linear portion of the bi-linear curve for each of the material directions of a ply. Unfortunately, it was discovered during this project that one of the three layers in the CDB200 triax fabric was actually a slightly different fabric than the others. After several attempts to resolve material properties, it was determined that the data was unusable. However, the results from the digital image correlation were quite encouraging and led to the the purchase of a dedicated camera and lens for future development of the system.

Boundary Displacements Glued Aluminum Tee-Joints

The most ongoing use of the DIC system has been with Aaron Sears on tests of aluminum T's with glued joints, shown in Figure 49. This project has been going through three generations of the DIC software development and shows the differences between them in it's results.

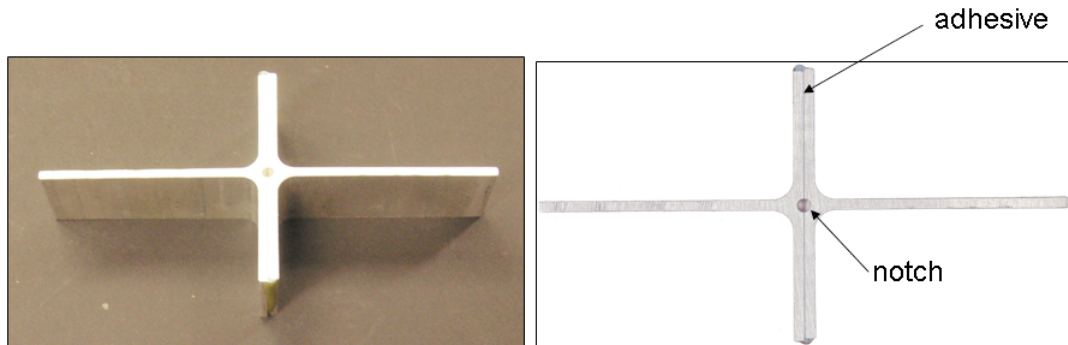


Figure 49: Aluminum T joint. Gage length of 47 mm, width of 25 mm, glue area of 16.1 cm^2

These joints were made using a high strength glue that needed to be characterized. These joints had a gage length of 47 mm and a width of 25 mm with a glue area of 16.1 cm^2 . An artificial stress concentration notch much like the notch on the composite IPL coupons was made with a drill to ensure that failure would occur at the same

spot on each test. The tension and shear response to failure of the glue was of interest for this project, and the IPL and the DED technique was seen as a possible way to characterize this response. These samples were very stiff and provided an interesting challenge for the DIC system because displacements would be quite small. They would also tax the abilities of the IPL because high loads would be produced and any problems with gripping and frame deflection would be apparent. These tests are included because they were done early in the development of the software side of the DIC system and show how its capabilities have improved over three generations of modifications.

The MATLAB scripts that were originally obtained from Eberl [22] were first used to try and correlate boundary displacements without modification. These scripts had been used successfully to find fiberglass coupon boundary displacements but never to track displacements as small as the aluminum T joints. The first results from this were not very promising. Figure 50 on the next page shows the original DIC boundary displacements compared to the LVDT displacements and a finite element model of the joint created in ANSYS by Aaron Sears.

These data show the response of the joint up until failure. The joint failed on the next load step and displacements jumped to 0.6 mm. This was left off this figure and the following figures for the sake of readability. During the last 3 steps before failure it was possible to see a crack forming on the glue joint near the stress concentration. This can barely be seen in the data as a decrease in stiffness over the last three points of the curve. After this testing it was apparent that there was room for improvement in the DIC system. After some time pouring over the MATLAB code and some trial and error, the problem was discovered to be a combination of a crippling modification of the MATLAB correlation script `cpcorr.m` on the part of Eberl and a limitation of 1/10th of a pixel set on the sub-pixel displacement calculation script `findpeak.m`.

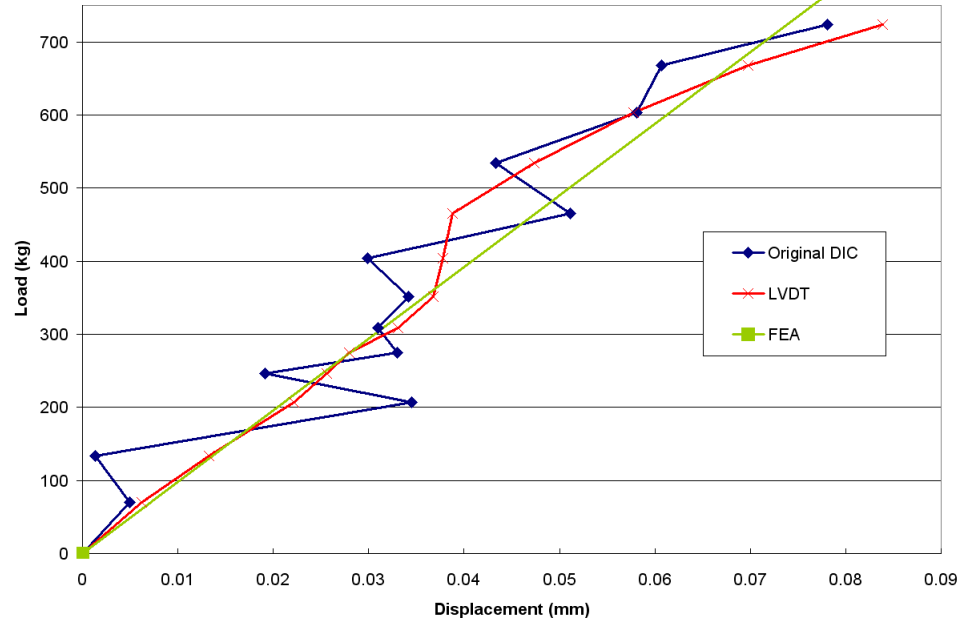


Figure 50: Boundary displacements in the y direction for a y test of an aluminum T joint as reported by the original DIC scripts, the LVDT's and as predicted by ANSYS

The modification was removed from `cpcorr.m` and the resolution of `findpeak.m` was changed to 1/1000 of a pixel and the correlation was run again. This time the results were much more promising, as seen in Figure 51 on the following page.

The second generation image correlation system provided results that appeared to be much better than the previous ones. There was still a small jog in the data though that suggests room for improvement. Stiffness matched the FEA model quite well. Table 8 on the next page shows how the stiffness of the load vs. displacement curve compares between the three generations of the DIC methodology as well as that of the LVDT data simulated FEM displacements. The stiffnesses were calculated by fitting a linear curve to all but the last 3 points of the curves where energy started dissipating as the glue pulled apart at the stress concentration. The curves were also forced to pass through the origin.

The third generation of the image correlation is the current method using the

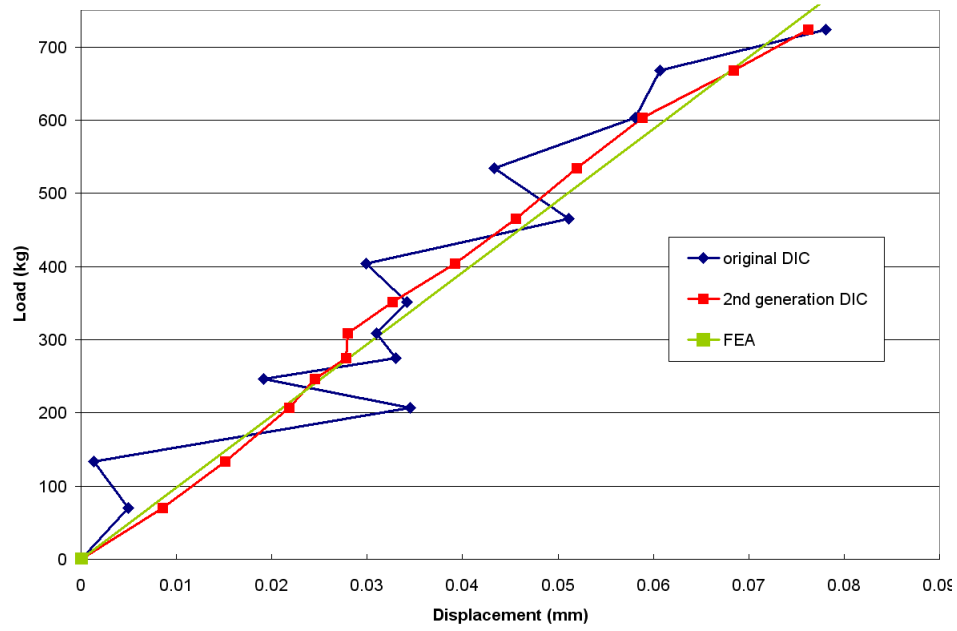


Figure 51: Boundary displacements in the y direction for a y test of an aluminum T joint as reported by the original DIC scripts, the LVDT's and as predicted by ANSYS

gradient method to calculate sub-pixel displacements. It can be seen compared to the second generation DIC and the FEM in Figure 52 on the following page.

The third generation DIC produces the most linear results out of all of them. This can be observed qualitatively but is also reflected as the lowest R^2 value out of the bunch in Table 8.

Table 8: Stiffness and R^2 Values

Measurement Method	Stiffness (kg/mm)	R^2 for Curve Fit
LVDT displacement	10,506	0.966
FEA model	9,800	1
Generation 1 DIC	10,016	0.765
Generation 2 DIC	10,248	0.9926
Generation 3 DIC	10,159	0.9976

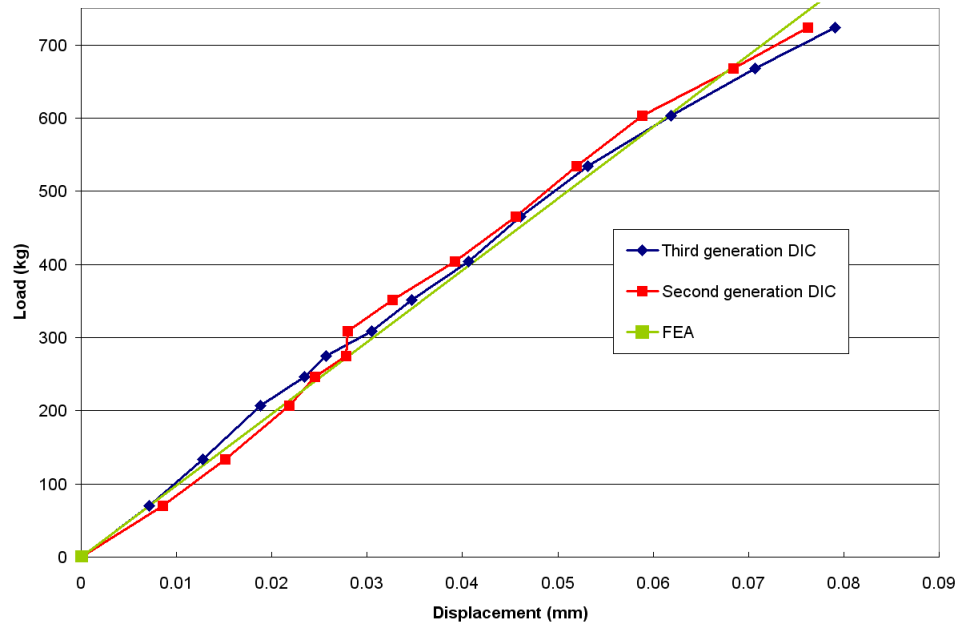


Figure 52: Boundary displacements in the y direction for a y test of an aluminum T joint as reported by the second and third generation DIC scripts and as predicted by ANSYS

Boundary Displacements and Full-Field Strain on Carbon Fiber and Epoxy Coupons

The most recent testing on FRP coupons using the DIC system was done by John Parker and James Schmitt in 2007/2008 as part of a project to develop a DED function for carbon fiber as mentioned in Chapter 2 on page 10. The coupons for this were made with IM7-G carbon fiber/ 8552 Epoxy resin unidirectional prepreg manufactured by Hexcel corporation laid up as $[30/-30]_s$. As was mentioned before seven load paths were chosen, which were:

- Pure tension, Y
- Pure shear, X
- Shear and Tension Y, X
- Tension and Closing Rotation $Y, -R$

- Shear and Opening Rotation Y, R
- Shear and Closing Rotation $X, -R$
- Shear, Tension and Opening Rotation X, Y, R

These tests were done with 50 steps each and images were captured at the completion of each step using an automated command line script executed by the IPL's Labview control software. Boundary displacements were calculated for all of these tests using the same boundary displacement script as before using the correlation coefficient peak-fitting technique.

While pure displacements were included most of the tests were done with combinations of displacements to provide a data rich environment for the DED function. Load combinations were chosen that would provide large variations in strain across the sample surface while reducing if not eliminating any compressive forces as it was assumed that out of plane deflection or buckling would result. There were some interesting interactions that could be seen on the data for these tests. Figure 53 on the next page shows load vs. displacement for test 5 which combined shear and tension (x and y).

The most notable difference between this and the previous DIC boundary displacement data is that there is considerably more detail due to the larger number of load steps that were imaged. It is critical to capture as much data as possible after damage first starts to accumulate for the development of the DED function so capturing this damage response is important. Each jump in the data can be seen on the images as a damage event, which eventually lead to large delamination and the damaged region extending all the way to the grips. Some of the tests displayed a more detailed damage response, Figure 54 on the following page shows the load vs. boundary displacement along the Y axis of test 1, which was a pure tension test.

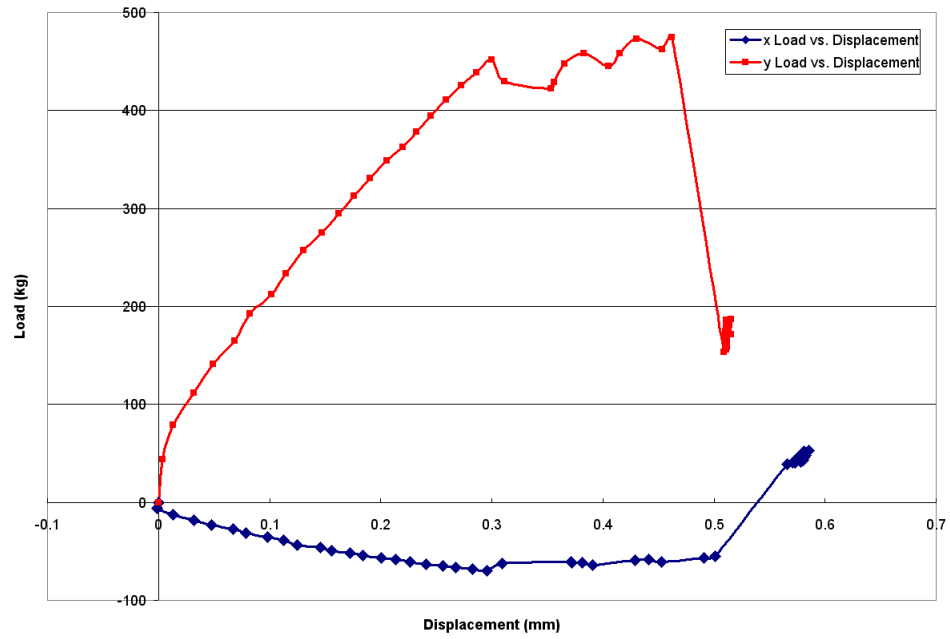


Figure 53: Boundary displacements in the x and y directions for a combined x and y test on a carbon/epoxy IPL coupon

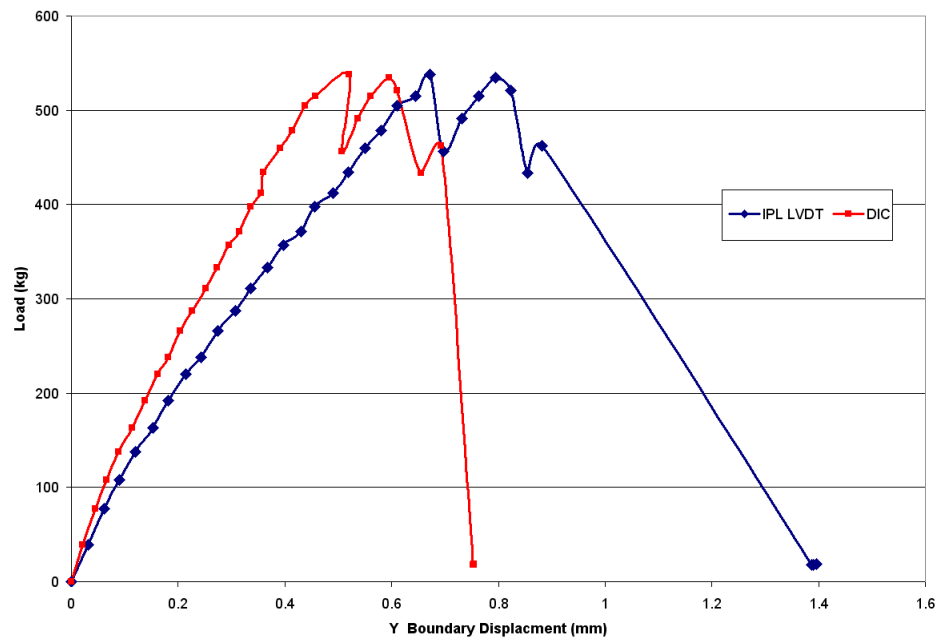


Figure 54: Boundary displacements in the y direction for a y test on a carbon/epoxy IPL coupon

In this figure it is possible to clearly see the reduction in stiffness as energy is dissipated through progressive damage eventually leading to total failure. Each jump down in force on the figure is due to a damage event occurring to the coupon, the final of which is catastrophic failure. This figure also shows how the DIC data compared to the LVDT boundary displacements. As can be seen, there is still a large difference at between the two at large loads.

The boundary displacements in this project were initially measured to be used as boundary conditions for the FEM model to artificially simulate strains for the DED function. However, with the ability to directly calculate strains from full field displacements this step is no longer necessary. Using strain data measured from the surface of the sample removes any error stemming from differences between the actual coupons and the FEM model that could negatively affect the development of the DED function. One interesting thing about these data is they show developing cracks in the samples very well. The following Figures 55 on the next page and 56 on the following page illustrate a developing crack as a concentration of ϵ_{yy} at the notch in the coupon.

The reduction in stiffness due to the formation of this crack can also be seen in Figure 54 on the previous page as the initial drop in load after the linear portion of the load vs. displacement curve. In this test the strain points around the crack were automatically removed after the paint chipped off the sample surface and the correlation was lost. Determining when the correlation data is no longer accurate is a problem that needs to be addressed when dealing with these full field behaviors. This issue is currently addressed by simply eliminating any strain points that fall outside of empirically determined upper and lower limits for strain. A system using a user defined mask has also been developed which allows the user to define which control points will have strain calculated for them before hand.

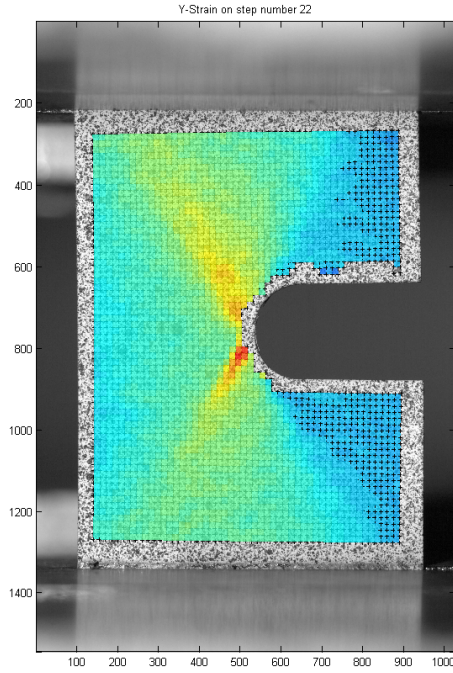


Figure 55: ϵ_y strain in a pure y displacement test just after damage initiation

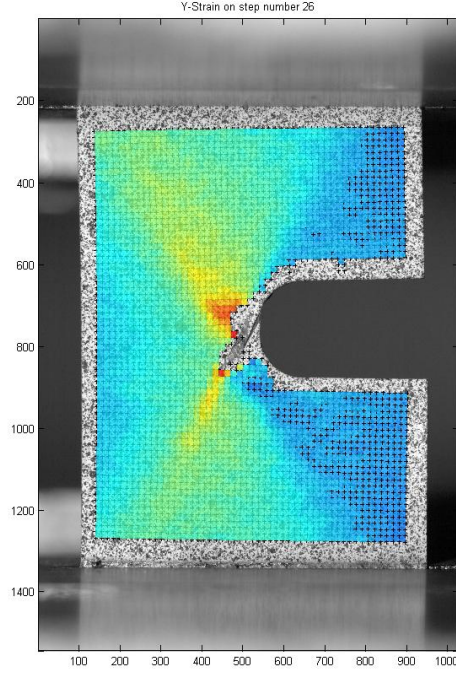


Figure 56: ϵ_y strain in a pure y displacement test with growing crack

Using DIC Strain for the DED function

As mentioned before, using the full-field strain data for the development of a DED function is the most promising use of the system developed for this thesis. There are a couple of issues that need to be overcome in order for this to happen. The first difficulty has already been addressed and is accurately determining the position of the strain field relative to the notch and boundaries of the coupon. When the grid is being generated for the test, the user is asked to select the top and bottom edges of the notch as well as the center, as seen in Figure 57 on the following page. This notch data is saved along with the displacements in a coordinate system that starts at the top-left corner of the gage section of the sample for later use. This location information can later be used for positioning the strain field accurately over the notch

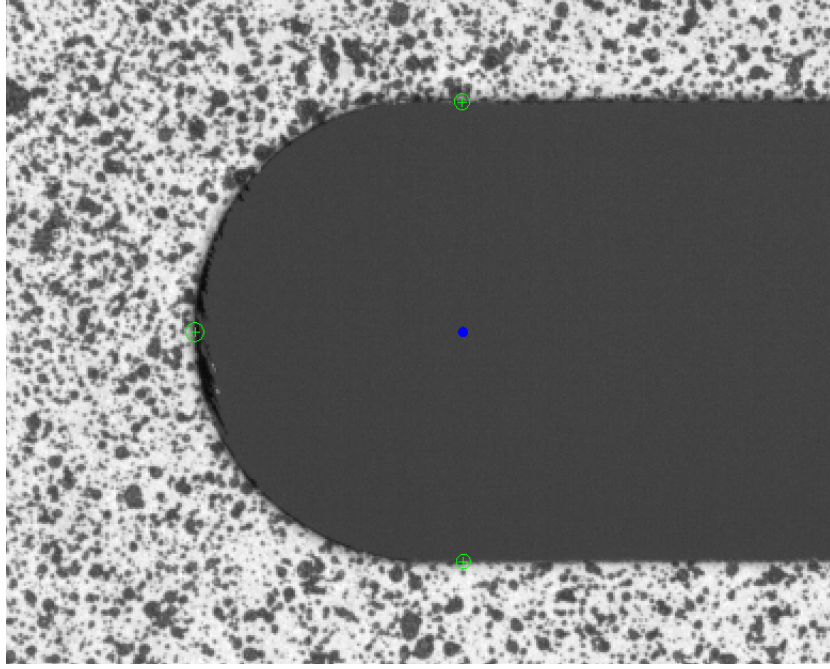


Figure 57: Selected points in the notch area to indicate their position for positioning the strain-field in development of the DED function

area. Another difficulty that is yet to be overcome is the lack of strain information on the edges of the sample in the notch area. When using the full field strain for the development of the DED function the high strains around the notch area not covered by the strain field are of the utmost importance, as they contain the largest strain gradient of the test, and much of the total strain data is in this area [40]. A method for calculating strains up to the boundary using forward or backward difference formulations for the partial derivatives would need to be used in order to calculate strains in this area. An example of a four-point forward difference formulation of $t_x(i, j)$ as found in [25] is as follows:

$$t(i, j) = \frac{-2t(i, j) + 9t(i + 1, j) - 18t(i + 2, j) + 11t(i + 3, j)}{6h}, \quad (26)$$

where h is the grid spacing and $t(i, j)$ is the value of the displacement matrix at point $x = i$ and $y = j$. A backward difference formulation would be simply pull points from

the displacement matrix in reverse, e.g,

$$t(i, j) = \frac{-2t(i, j) + 9t(i - 1, j) - 18t(i - 2, j) + 11t(i - 3, j)}{6h}. \quad (27)$$

As before, the formulations for t_y would be similar only the points used will be chosen along the y direction instead of the x . It is possible that a four-point formulation would be averaging over too large of an area without changing the control point grid spacing of the points in the notch area, better results could be obtained from a three or two point formulation. As was mentioned in section 3 on page 26 the error in the strain calculation is a complicated affair and warrants further investigation. These forward and backwards difference formulations were not used in this thesis, but if they were to be incorporated into the MATLAB scripts it would be done as follows:

1. First the control point grid would need to be refined around the notch so that there are grid points along the edges wherever possible. This could be done manually, although a method that uses the positions of the notch points as indicated by the user in the grid generation process could be made that would automatically place points around the curve of the notch. Since the control points indicate the center of the sub images as mentioned in chapter 3 on page 21 there would be overlap off the edge of the sample, but as long as the background does not contain any detail it should not contribute negatively to the peak of the correlation function.
2. After the grid was defined it would need to be determined which finite difference formulation would be used for each boundary control point. This could also be done automatically by a simple algorithm that follows the boundary of the control-point grid and calculates forward and backward differences were appropriate using grid points from within the boundaries. It could also be

possible to take advantage of the lack of detail on the parts of sub-images that overlap the notch boundary and use a Fourier transform in a script that could determine how much a sub-image overlapped the sample by the amount of high-frequency information in its transform. This script could then be used to automatically determine where the points are located relative to the notch and then perform the appropriate approximation of the derivative.

The current strain calculation algorithm is written for a rectangular set of control points, so the above additions would require modifying a few assumptions made within the MATLAB scripts. However, the result would be a program that can show strain data up to the boundaries of the sample, which is something even the commercial DIC package VIC-2D does not do (more information on VIC-2D in [44]).

IMPLICATIONS FOR COMPOSITE STRUCTURES CHARACTERIZATION

The DIC technique as developed for this thesis has some interesting implications for the characterization of composite structures, most notable of which are for the determination of non-linear constitutive properties and the development of a DED function for a laminate.

Non-Linear Constitutive Response

The FE modeling of a non-linear constitutive response using IPL test data was most recently done by James Shchmitt in [40] with a bilinear response curve. Instead of giving each element of the FE model a linearly increasing stiffness until failure, they are given a stiffness that is linear up to a critical strain, ϵ_{11c} , and then linearly reduced in stiffness back to zero stiffness at an ultimate strain of ϵ_{11u} . A bilinear response is shown in Figure 58 on the next page along with two different strainfields on an IPL sample. Full field strain can be used to directly measure strains off of the sample during damage events; these data can then be used to create a model that more accurately matches strains seen in testing. This would again provide self-consistency to the process by capturing various ply-interactions that can affect behavior that are not captured when using only boundary displacements. The term self consistent used here means that the same tools used to generate the nonlinear constitutive response and dissipated energy model (i.e. finite element analysis) can then be used for subsequent analysis of composite structures independent of geometry. With this self consistent approach a designer has some assurance that the predicted structural response is directly related to the experimental database for progressive damage. The

resulting FE model will then ostensibly model the response of composite structures more accurately.

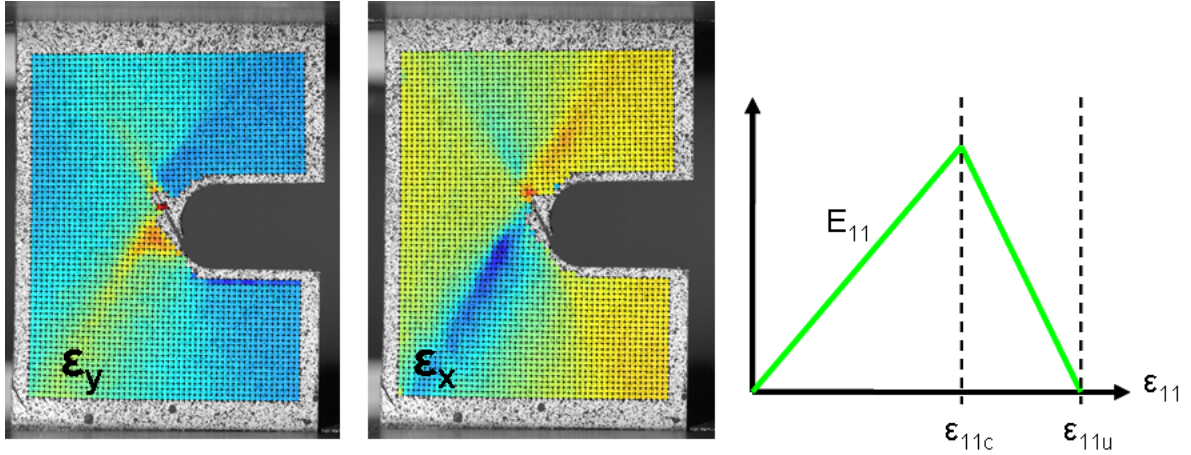


Figure 58: ϵ_y and ϵ_x strain on an IPL sample and a bilinear constitutive response

Full-Field Strain and Dissipated Energy

The use of full-field strain data for the DED technique shows potential to both simplify and enhance this method. Incorporating full field strain information into the DED technique eliminates the need for a FEA strain model to interpret the progressive damage data and increases the accuracy of the data used to develop the DED interpolation function. A flow chart of the simplified DED process can be seen in Figure 59 on the following page. In this manner the Finite Element Method is used to build the non-linear constitutive model as opposed to interpret the data and strainfields. Again, this builds on the idea of self consistency as described above.

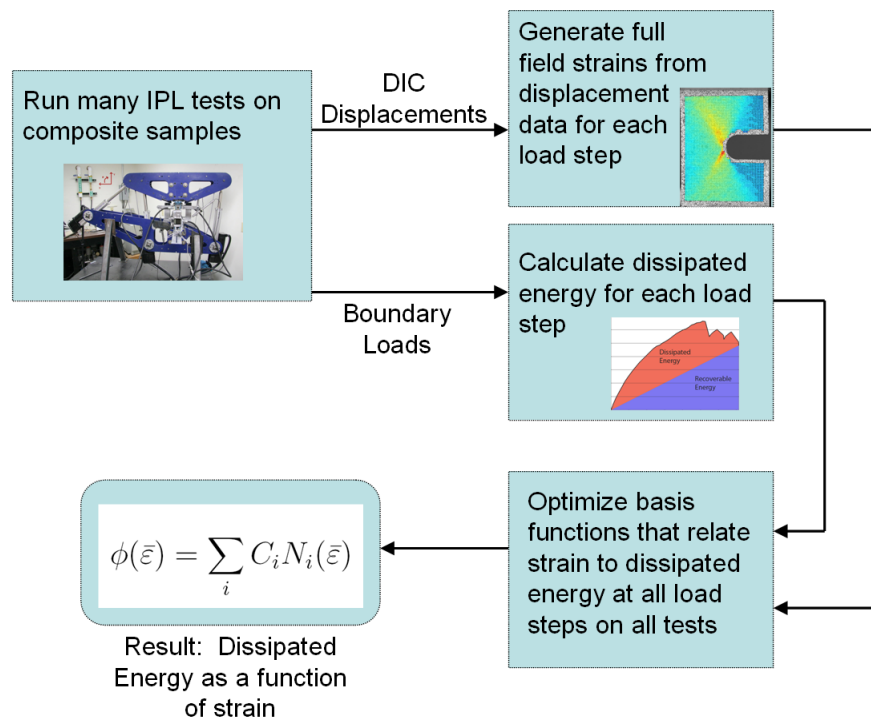


Figure 59: How the DIC system is used to develop the DED function using full field strain

Other Implications

The DIC system is quite flexible in how it can be implemented and can be used for testing beyond incorporation into the DED technique.

The DIC system developed in this thesis could have some interesting implications for the characterization of composite structures. The camera system itself is quite portable, with an even more portable system possible. Images can be taken of many different components of a structure during testing, or even in the field during use or under some kind of proof load. Using the DED method, strain from these images can be used to predict the level of damage that has occurred and predict the residual strength. Complicated structures can be viewed from different angles, any flat surface that the camera can focus on can have displacements measured on it.

The software used in this thesis is very flexible as it is all written as MATLAB scripts and functions, and it can easily be modified to serve a particular purpose. Boundary displacement calculations have been done already. There are many other possible uses. One such use could be a system that would detect and monitor crack growth on a fatigue specimen by measuring the displacements of control points on either side of the crack at fixed cycle intervals, or even every cycle. Another use could be a portable system that can be mounted on a structure during use and record image that will reveal strains on a surface during actual service in the field.

CONCLUSION AND FUTURE WORK

Future Work

There are a few areas that warrant further investigation in this project. The first item that would be worth improving is imaging speed. The current command line program that triggers the camera to take a picture is far from speedy. There is a few second delay from when the computer tells the camera to shoot and the actual shutter release. There is then a minute of time when the camera is unusable and flashing a “busy “ message on the LCD. The reason for this is unknown, and it is present in all cases when using software developed with the Canon software development kit as well as Canon’s own control software, EOS capture. As mentioned before, DSLR Remote Pro produced by Breeze Systems has been used successfully and is significantly faster. Another option would be to use the PTP Camera Toolkit, which is a collection of LabView programs written by Hytek Automation. These scripts control the camera directly from LabView without having to go through a command line program as was done for this thesis. This would be a good option for integrating the image correlation into the IPL testing procedure as a part of the same LabView program that controls the IPL.

Another intriguing concept is the use of an even less expensive camera for testing. A digital SLR was used for this project primarily because of it’s RAW file output and high image quality. However, there are a few point and shoot cameras out there that have RAW capability and high quality optics such as the canon S60, S70 or G9. There are others made by other brands as well. Another option would be using the Canon firmware add-on CHDK which gives almost any inexpensive Canon camera RAW capability as well as advanced control over shutter timing. Such a system could

likely improve testing speed at a lower cost at the expense of accuracy. How much accuracy would be lost would have to be investigated.

If more accuracy is desired, there is another sub-pixel method that could be implemented. This method is used by the commercial DIC package VIC-2D and is popular in the literature researched for this thesis as well. In this method a new sub-image is interpolated with a sub-pixel shift in displacement and then correlated to the base-image. An iterative optimization technique such as the Newton-Raphson or Levenberg-Marquardt method is then performed to maximize calculated correlation value. This method is covered in [35] and [41], the latter of which compares various interpolation schemes. It has been found by [35] to provide more accurate results than the two methods used in this thesis at the expense of calculation time. Coding for a function based on this method using MATLAB's non-linear optimization function *lsqnonlin* and the 2-D interpolation function *interp2* was started for this thesis but needs to be finished before any results can be produced.

Future Work, using full-field strain data

There is a potential complication to using full-field data for modeling a constitutive response or for a DED function that needs to be overcome. This is due to the macroscopically non homogeneous nature of composite materials coupled with the notched geometry of the IPL coupons. The notch forces high stresses over a small area on the coupon. The strength of any given small area on different coupons with the same layup and manufacturing can vary greatly due to flaws and the fiber and matrix quantities, and other inhomogeneities that are present. These coupons can exhibit a dramatically different constitutive response when looking in detail over the entire surface while at the same time showing a similar response when only looking at

the boundaries. This issue could potentially be overcome by including many different tests in order to regain some of the quantitative statistics lost when looking at failure in such detail, or even just decreasing the amount of detail captured by increasing grid spacing. However, this complication may yield important data for probabilistic design methodologies that are concerned with statistical variation in the properties of a material. These variations are not accurately captured by traditional continuum progressive damage modeling.

Future Work, improving IPL results with DIC

There are several ways in which the DIC system can be further integrated into the IPL test procedure to improve its capabilities. Perhaps the most useful way that the DIC system could improve the IPL results is to integrate it as a control system. The current system uses LVDT grip displacements and a feedback loop to adjust actuator displacements until the grips match the desired displacements. A similar system could be made that would use real time DIC to track a few points on the boundaries and use these boundary displacements to control the IPL. Implementing this would require downloading the images to the IPL control computer during testing, which would require a different method of camera control. The CHDK software mod mentioned earlier could potentially be good for this application, as it can provide exact shutter timing and RAW output to a inexpensive dedicated control system camera that could be fully integrated into the IPL.

Improving the IPL itself

After spending several years working with the IPL, it has become clear that it would be much more capable if some base level improvements were made to its structure. Increasing the stiffness of the frame and grips would significantly increase its functionality for use in composite material characterization. When gathering data for the DED function the post damage response of the material as it dissipates energy is most important. With very stiff samples, the frame and grip assembly of the IPL can be significantly more compliant than the sample itself. In these cases, elastic energy being stored before failure as IPL frame compliance is released as a large boundary displacement on the sample when it starts to accrue damage and lose strength. Much work has been done for this thesis and by Collett and Smith [17] [42] to compensate for excessive compliance in the frame and grips of the IPL. These problems are more difficult to manage than the gripping effects, which are effectively removed through the use of DIC. Rather than compensate for this unwanted compliance, it could be addressed directly by increasing the frame and grip stiffness of the IPL. The majority of the system compliance is in the grip assembly. If a DIC displacement feedback system were developed the grip mounted LVDT system could be removed which would allow more room for robust grips as LVDT clearance was a major design obstacle with the current configuration. The self-centering grip system, while useful, could also be replaced with a simple and more robust system that can manually be adjusted for different coupon thicknesses. The IPL is generally used for a series of many tests with the same coupon geometry and thickness to produce data for a DED function, so having to adjust to a different coupon thickness would only be a one-time procedure for each new material being tested.

Conclusion

This project has demonstrated that full field displacement and strain data can be obtained from the surface of a planar test sample successfully using an off the shelf consumer camera and lens and MATLAB functions that are available without cost. Results from uniaxial and multiaxial testing of metals and composite materials have been shown the system to be comparable to that of much more expensive commercial DIC systems, and it's implementation much more flexible. The code can be modified to a specific purpose, such as the boundary displacement tracking script developed for this thesis. Many other uses are possible that could tailor the system to all kinds of testing situations, and with the low cost of the camera equipment, implementation is possible for small projects and budgets as long as MATLAB is available, which makes it quite viable in academics.

REFERENCES CITED

- [1] ABRAMOWITZ, M. and STEGUN, I. A., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. New York: Dover, ninth dover printing, tenth gpo printing ed., 1964.
- [2] ALLIANCE, A. C., "History of automotive composites."
Available at: <http://www.autocomposites.org/composites101/history.cfm>, 2008.
- [3] ASKEY, P., "Canon eos 300d / digital rebel / kiss digital review."
Available at: <http://www.dpreview.com/reviews/canoneos300d/>, 2003.
- [4] ASKEY, P., "Canon eos 350d / digital rebel xt/ kiss n digital review."
Available at: <http://www.dpreview.com/reviews/CanonEOS350D/page27.asp>, 2005.
- [5] ASTM, "Classification of extensometer systems," tech. rep., ASTM, 2006.
- [6] BARBERO, E. J., *Introduction to Composite Materials Design*. Taylor and Francis, 1998.
- [7] BAYER, B. E., "Color imaging array." U.S. Patent 3,971,065, 1975.
- [8] BOEING, "Boeing logs 12-hour a160t hummingbird flight."
Available at: http://www.boeing.com/ids/news/2007/q4/071016a_pr.html, 2007.
- [9] BOEING, "F/a-18 hornet fighter: History."
Available at: <http://www.boeing.com/history/mdc/fa-18.htm>, 2008.
- [10] BOEING, "Fast facts: Boeing 787."
Available at: <http://www.boeing.com/news/feature/sevenseries/787.html>, 2008.
- [11] BREEZE SYSTEMS, "Dslr remote pro - remote capture software."
Available at: <http://www.breezesys.com/DSLRRemotePro/index.htm>, 2008.
- [12] BUDINSKI, K. G. and BUDINSKI, M. K., *Engineering Materials: Properties and Selection*. Prentice Hall, 1999.
- [13] CANON, "Canon full frame CMOS sensors," *Canon White Paper*, 2006.
- [14] CANON, "Infrared-cut low-pass filters," *Canon Technology*, 2008.
- [15] CANON USA INC., "Canon digital camera software developers kit."
Available at: http://www.usa.canon.com/consumer/controller?act=SDKHomePageAct&keycode=Sdk_Lic&fcategoryid=314&modelid=7474&id=3464, 2008.

- [16] COFFIN, D., “Decoding raw digital images in linux.”
Available at: <http://www.cybercom.net/~dcoffin/dcraw/>, 2008.
- [17] COLLETT, A., “A validation study of the montana state university in-plane loader,” Master’s thesis, Montana State University Department of Mechanical and Industrial Engineering, 2006.
- [18] DAVIS, C. Q. and FREEMAN, D. M., “Statistics of subpixel registration algorithms based on spatiotemporal gradients or block matching,” *Optical Engineering*, vol. 37, pp. 1290–12981, 1998.
- [19] DAVIS, C. Q., FREEMAN, D. M., and KARU, Z. Z., “Equivalence of subpixel motion estimators based on optical flow and block matching,” in *Proceedings of the International Symposium on Computer Vision*, pp. 7–12, 1995.
- [20] DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF AUCKLAND, “JPEG: joint photographic experts group,” *Spatial and Spectral encoding, University of Auckland Department of Computer Science*.
- [21] EASTMAN KODAK, “New kodak image sensor technology redefines digital image capture,” tech. rep., Eastman Kodak, 2007.
- [22] EBERL, C., “Digital image correlation and tracking with matlab.”
Available at: <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=12413&objectType=File>, 2007.
- [23] ERIC BOOTH, M. S., “In-plane loader, a multi-axis composite testing machine.” Senior Design Project, 2001.
- [24] FOVEON INC., “Foveon X3® Pro 10M CMOS Direct Image Sensor.”
Available at: http://www.foveon.com/files/Product_FoveonF7_050404_P.pdf, 2004.
- [25] GRIFFITHS, D. V. and SMITH, I. M., *Numerical Methods for Engineers*. CRC Press, 2006.
- [26] HARA, Y., “Canon uses CMOS sensor to gain digicam edge,” *EE times online*, 2000.
- [27] HUANG, D. and CAIRNS, D. S., “Constitutive laws of composite materials via multiaxial testing and numerical optimization,” *Journal of Reinforced Plastics and Composites*, no. 3, 2006.
- [28] JONES, R. M., *Mechanics of Composite Materials*. Taylor and Francis, 1999.
- [29] LEWIS, J. P., “Fast normalized cross correlation,” *J. P. Lewis Research Archive*, 1995.

- [30] MAST, P. W., NASH, G. E., MICHPOULOS, J., THOMAS, R. W., BADALIAN, R., and WOLOCK, I., "Experimental determination of dissipated energy density as a measure of strain-induced damage in composites," tech. rep., Naval Research Laboratory, 1992.
- [31] MATHEWS, J. H. and FINK, K. K., *Numerical Methods Using Matlab*. Prentice Hall, 2004.
- [32] NATIONAL INSTRUMENTS, "NI labview."
Available at: <http://www.ni.com/labview/>, 2008.
- [33] NETWORK, L. C. O. G. T., "Bright stars/objects, blooming and residual "ghost" images."
Available at: <http://faulkes-telescope.com/education/planning/ghosting/>.
- [34] OLEXA, R., "F-35: Flying lean," *Manufacturing Engineering*, vol. 130, 2003.
- [35] PAN BING, XIE HUI-MIN, E. A., "Performance of sub-pixel registration algorithms in digital image correlation," *Measurement Science and Technology*, pp. 1615–1621, 2006.
- [36] PHOTOZONE.DE, "Tokina af 100mm f/2.8 at-x pro d macro - review / test report," *Photozone Reviews*, 2006.
- [37] RAMANATH, R., SNYDER, W., BILBRO, G., and SANDER, W., "Demosaieking methods for bayer color arrays."
Available at: <http://citeseer.ist.psu.edu/article/ramanath02demosaieking.html>, July 2002.
- [38] RITTER, W., "Application of energy methods to modeling failurs in composite materials and structure," Master's thesis, Montana State University Department of Mechanical and Industrial Engineering, 2004.
- [39] RUSSEL, D. J., "Market trends: The composites affordability initiative, part i," *Composites world*, 2007.
- [40] SCHMITT, J., "Damage initiation and post-damage response of composite laminates by multi-axial testing and nonlinear optimization," Master's thesis, Montana State University Department of Mechanical and Industrial Engineering, 2008.
- [41] SCHREIER, H. W., BRAASCH, J. R., and SUTTON, M. A., "Systematic errors in digital image correlation caused by intensity interpolation," *Optical Engineering*, vol. 39, pp. 2915–2921, 2000.

- [42] SMITH, J., "Internal damage characterization for composite materials under biaxial loading configuration," Master's thesis, Montana State University Department of Mechanical and Industrial Engineering, 2007.
- [43] SMOCK, D., "Boeing 787 dreamliner represents composites revolution." Available at: <http://www.designnews.com/article/CA6441583.html>, 2007.
- [44] SOLUTIONS, C., "Non-contacting measurement solutions." Available at: http://www.correlatedsolutions.com/index.php?option=com_frontpage&Itemid=1.
- [45] STRONG, A. B., "The history of composite materials, oppurtunities and necessities." Available at: <http://www.et.byu.edu/groups/strong/pages/articles/articles/history.pdf>.
- [46] THE MATHWORKS, INC., "Matlab image processing." Available at: <http://www.mathworks.com/applications/imageprocessing/>, 2008.
- [47] THE OWENS CORNING COMPANY, "A history of innovation." Available at: <http://www.owenscorning.com/acquainted/about/history/index.asp>, 2008.
- [48] TSAI, S. W. and WU, E. M., "A general theory of strength for anisotropic materials," *Journal of Composite Materials*, no. 1, 1971.
- [49] WARNER ELECTRIC, "Electric actuator and actuator control line from warner linear." Available at: <http://www.warnerelectric.com/newproducts.asp?category=actuators>, 2008.
- [50] W.H. PETERS, W. R., "Digital imaging techniques in experimental stress analysis," *Optical Engineering*, vol. 21, pp. 427–431, 1982.
- [51] ZHANG, D. and AROLA, D. D., "Applications of digital image correlation to biological tissues," *Journal of Biomedical Optics*, no. 4, 2004.

APPENDICES

APPENDIX A

ASTM EXTENSOMETER REPORT

EXTENSOMETER VERIFICATION REPORT FOR A DIGITAL IMAGE CORRELATION BASED SYSTEM

EXTENSOMETER Strain generated with MATLAB scripts JP_inplanestrains_centraldiff.m from displacements calculated by CPCORR_gradient.m

STRAIN: Strains generated using MATLAB image transformation and piece-wise bilinear interpolation using a full size image from a standard IPL coupon test sequence. This is a perfect situation and only demonstrates the sensitivity of the software system, the image acquisition and processing will add it's own error in real-world tests.

TEST TYPE: Unidirectional Temperature:N/A

GAGE LENGTH: 1.095in (27.8 mm) measured off of image. For this DIC technique gage length is not used for the calculation of strains, the control point spacing is. For this reason, two different control point spacings were used, one was 50 pixels and the other was 15 pixels.

GAGE LENGTH ERROR: Max of 0.14%

Actual Strain %		Extensometer Strain %		Fixed error pixel/pixel		Relative Error % of act strain	
Run 1	Run 2	Run 1	Run 2	Run 1	Run 2	Run 1	Run 2
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
0.2500	0.2500	0.2490	0.2515	0.0011	0.0015	0.4188	0.6034
2.5000	2.5000	2.1971	2.4922	0.0029	0.0015	0.1162	0.3129
5.0000	5.0000	4.9851	4.9857	0.0149	0.0078	0.2986	0.2867
7.5000	7.5000	7.4835	7.4786	0.0165	0.0143	0.2204	0.2855
10.000	10.000	9.9760	9.9721	0.0240	0.0279	0.2396	0.2786
Run 1: 15 Pixel Control Point Spacing				Run 2: 50 Pixel Control Point Spacing			
Resolution: 1×10^6 pixels				Class of Extensometer system: B-2			

VERIFICATION METHOD: Digitally strained images

VERIFICATION APPARATUS: Matlab Software

Verification Performed Per ASTM Standard Practice E83-06

APPENDIX B

MATLAB CODE

Here is the Matlab Code for running the correlation in the order in which it is typically used. Auxillary code and the modified MATLAB functions used for image correlation are also included.

JP_IPL_strain_grid_generator.m

This function generates grid points for either a evenly spaced full field strain grid, or just two horizontal top and bottom boundary grids.

```
%To use this function follow the directions given and
%select the bottom left,
%top right and notch points and then have it fit a grid onto
it for strain
%field generation OR just generate top and bottom rows of grid
points to
%do boundary displacement correlation.
```

```
FileNameBase = uigetfile( ...
    {'*.bmp;*.tif;*.jpg;*.TIF;*.BMP;*.JPG','Image_files...
    '*.bmp;*.tif;*.jpg','*.*', 'All_files_(*.*)'}, ...
    'Open_base_image_for_grid_creation');

%This was for saving the boundary points for other
applications, uncomment
%if needed
% points(1,1:2)=topleft(1:2);
% points(2,1:2)=topright(1:2);
% points(3,1:2)=bottomleft(1:2);
% points(4,1:2)=bottomright(1:2);
% save points.mat points

hfig = figure('Menubar', 'none',...
    'Name','select_bottom_left,top_right,notch_top
    ,...
    notch_bottom(near_inside),notch_center',
    'NumberTitle','off',...
    'IntegerHandle','off');
im_grid = imread(FileNameBase);
himage = imshow(im_grid);
% Add the scroll panel.
hpanel = imscrollpanel(hfig,himage);
% Position the scroll panel to accommodate the other tools.
set(hpanel,'Units','normalized','Position',[0 .1 1 .9]);
% Add the Magnification box.
hMagBox = immagbox(hfig,himage);
%pixel info tool
hpixinfo = impixelinfo(himage);

%input points, start with bottom left
[bottomleft2(1,1:2)]=ginput(1);
hold on
plot(bottomleft2(1,1),bottomleft2(1,2),'+g')
```

```

[topright2(1,1:2)]=ginput(1);
hold on
plot(topright2(1,1),topright2(1,2),'+g')

[notcht2op(1,1:2)]=ginput(1);
hold on
plot(notcht2op(1,1),notcht2op(1,2),'+g')

[notchbottom(1,1:2)]=ginput(1);
hold on
plot(notchbottom(1,1),notchbottom(1,2),'+g')

%helps you locate the notch center based on top and bottom of
    the notch
notch=(notcht2op(1,2)+notchbottom(1,2))/2;
plot (notchbottom(1,1),notch,'*b');

%input notch center point
[notchcenter(1,1:2)]=ginput(1);
hold on
plot(notchcenter(1,1),notchcenter(1,2),'+g')

% grid resolution. Strainfield will use this as an initial
    value but
% change it so that the far edges will be covered. 10 and 15
    pixels values
% seem to work well here.
xspacing = 15;
yspacing = 15;

%decide whether to save a strain grid or boundary grid. You
    can turn a
%boundary grid into a strain grid with
    JP_boundary_grid_to_strain_grid.m
%later if you want to do both
gridselection = menu(sprintf('What do you want?'),...
    'strainfield_grid','boundary_displacement_grid');
if gridselection==1

    %this is the regular spaced grid for strainfield use
    %distances. Starts by making topleft and topright points
        from the other
    %two that were originally selected
    topleft2=[bottomleft2(1,1) topright2(1,2)];
    bottomright2=[topright2(1,1),bottomleft2(1,2)];
    plot(bottomright2(1,1),bottomright2(1,2),'+y',...
        topleft2(1,1),topleft2(1,2),'+y');

    xdists=topleft2(1,1)-topleft2(1,1);

```

```

ydist=bottomright2(1,2)-topright2(1,2);

[x,y] = meshgrid(topleft2(1,1):xspacing:topright2(1,1)...
    , topleft2(1,2):yspacing:bottomleft2(1,2));
plot(x,y,'+b')

grid_x=x;
grid_y=y;

end
if gridselection==2
    %this is the grid with 2 rows, one top one bottom for
    boundary
    %displacements
    msgbox('click_top_left_corner_followed_by_bottom_right')

    [topleft2(1,1:2)]=ginput(1);
hold on
plot(topleft2(1,1),topleft2(1,2),'+g')

    [bottomright2(1,1:2)]=ginput(1);
hold on
plot(bottomright2(1,1),bottomright2(1,2),'+g')

x1 = topleft2(1,1):xspacing:topright2(1,1);
x1(1,size(x1,2))=topright2(1,1); %makes sure last point is
    exactly spaced

%the following makes sure both rows have the same amount of
    points

%if size(bottomleft2(1,1):xspacing2:bottomright2(1,1),2)~=size
    (x1,2)-1
xspacing2=(bottomright2(1,1)-bottomleft2(1,1))/(size(x1,2)-1);
xspacing2=abs(xspacing2);
x2 = bottomleft2(1,1):xspacing2:bottomright2(1,1);
%w2=size(x2,2);
%x2(1,w2+1)=bottomright2(1,1); %makes sure last point is
    exactly spaced
% else
%     end
x=[x1;x2];

%Adjusts Y spacing so that the two rows are equal distances
    from the notch
%it compares the locations of all 4 points first and picks
    the two that
%are closest to begin with. Commented for testing other things
    than ipl

```

```
%coupons
```

```
%
% diff(1)=dist1-dist2;
% diff(2)=dist3-dist4;
% diff(3)=dist1-dist4;
% diff(4)=dist3-dist2;
%
% if diff(1)<diff(2)
%     diff5=diff(1);
%     dist=mean(dist1,dist2);
%     else
%     diff=diff(2);
%     dist=mean(dist3,dist4);
% end
% if diff(3)<diff(4)
%     diff6=diff(3)
%     dist=mean(dist1,dist4);
% else
%     diff
%     dist
% end
% if diff(4)<diff5
```

```
% diff(1)=dist1-dist2;
% diff(2)=dist3-dist4;
% if diff(1)<diff(2)
%     dist=min(dist1,dist2);
%     else
%     dist=min(dist3,dist4);
% end
```

```
% dist(1)=notch-topright2(1,2);
% dist(2)=bottomleft2(1,2)-notch;
% dist(3)=notch-topleft2(1,2);
% dist(4)=bottomright2(1,2)-notch;
% dist=min(dist);
% y1=ones(1,size(x1,2))*(notch-dist);
% y2=ones(1,size(x2,2))*(notch+dist);
%uncomment above and the dist1,2,3,4 lines to have equally
% spaced grid
%points
```

```
%all y values are the same, uses the left side coordinates
```

```
y1=ones(1,size(x1,2))*topleft2(1,2);
y2=ones(1,size(x2,2))*bottomleft2(1,2);
```

```
y=[y1;y2];
```

```

plot(x,y,'+b')

grid_x=x;
grid_y=y;
%assembles notch points into a matrix and changes their
coordinates to
%originate at the top left corner
notchpoints(1,1:2)=notchtop;
notchpoints(2,1:2)=notchbottom;
notchpoints(3,1:2)=notchcenter;

notchpoints(:,1)=notchpoints(:,1)-topleft2(1);
notchpoints(:,2)=notchpoints(:,2)-topleft2(2);

save notchpoints.dat notchpoints -ascii -tabs
end

% Do you want to keep the grid?
confirmselection = menu(sprintf('Do you want to use this grid?
'),...
    'Yes','No');

if confirmselection==1
% Save settings and grid files in the image directory for
%visualization/plotting later
% save settings.dat xspacing yspacing
%amin_new xmax_new ymin_new ymax_new -ascii -tabs
    save grid_x.dat x -ascii -tabs
    save grid_y.dat y -ascii -tabs
    hold off
    close all
    clear x* y* t* b* i* himage hpanel hpixinfo hfig hmagbox
    clear notchtop notchbottom
end

if confirmselection==2
    close all
    hold off
% imshow(im_grid,'true size');
% IPL_grid_generator(FileNameBase);
end

```

filelist_generator.m

This function generates a list of image files for correlation. File names should end with sequential numbers in the order that they are to be correlated.

*%To use this function follow the directions given and
%select the bottom left,*

```

%top right and notch points and then have it fit a grid onto
    it for strain
%field generation OR just generate top and bottom rows of grid
    points to
%do boundary displacement correlation.

FileNameBase = uigetfile( ...
    {'*.bmp;*.tif;*.jpg;*.TIF;*.BMP;*.JPG','Image_files...
    '*.bmp;*.tif;*.jpg;*.TIF;*.BMP;*.JPG','All_Files_(*.*)'}, ...
    'Open_base_image_for_grid_creation');

%This was for saving the boundary points for other
    applications, uncomment
%if needed
% points(1,1:2)=topleft(1:2);
% points(2,1:2)=topright(1:2);
% points(3,1:2)=bottomleft(1:2);
% points(4,1:2)=bottomright(1:2);
% save points.mat points

hfig = figure('Menubar','none',...
    'Name','select_bottom_left,top_right,notch_top
    ,...
    ,notch_bottom(near_inside),notch_center',
    'NumberTitle','off',...
    'IntegerHandle','off');
im_grid = imread(FileNameBase);
himage = imshow(im_grid);
% Add the scroll panel.
hpanel = imscrollpanel(hfig,himage);
% Position the scroll panel to accommodate the other tools.
set(hpanel,'Units','normalized','Position',[0 .1 1 .9]);
% Add the Magnification box.
hMagBox = immagbox(hfig,himage);
%pixel info tool
hpixinfo = impixelinfo(himage);

%input points, start with bottom left
[bottomleft2(1,1:2)]=ginput(1);
hold on
plot(bottomleft2(1,1),bottomleft2(1,2),'+g')

[topright2(1,1:2)]=ginput(1);
hold on
plot(topright2(1,1),topright2(1,2),'+g')

[notchtop(1,1:2)]=ginput(1);
hold on
plot(notchtop(1,1),notchtop(1,2),'+g')

[notchbottom(1,1:2)]=ginput(1);
hold on
plot(notchbottom(1,1),notchbottom(1,2),'+g')

```

```

%helps you locate the notch center based on top and bottom of
    the notch
notch=(notchtop(1,2)+notchbottom(1,2))/2;
plot (notchbottom(1,1),notch,'*b');

%input notch center point
[notchcenter(1,1:2)]=ginput(1);
hold on
plot(notchcenter(1,1),notchcenter(1,2),'+g')

% grid resolution. Strainfield will use this as an initial
    value but
% change it so that the far edges will be covered. 10 and 15
    pixels values
% seem to work well here.
xspacing = 15;
yspacing = 15;

%decide whether to save a strain grid or boundary grid. You
    can turn a
%boundary grid into a strain grid with
    JP_boundary_grid_to_strain_grid.m
%later if you want to do both
gridselection = menu(sprintf('What do you want?'),...
    'strainfield_grid','boundary_displacement_grid');
if gridselection==1

    %this is the regular spaced grid for strainfield use
    %distances. Starts by making topleft and topright points
        from the other
    %two that were originally selected
    topleft2=[bottomleft2(1,1) topright2(1,2)];
    bottomright2=[topright2(1,1),bottomleft2(1,2)];
    plot(bottomright2(1,1),bottomright2(1,2),'+y',...
        topleft2(1,1),topleft2(1,2),'+y');

    xdist=topright2(1,1)-topleft2(1,1);
    ydist=bottomright2(1,2)-topright2(1,2);

    [x,y] = meshgrid(topleft2(1,1):xspacing:topright2(1,1)...
        ,topleft2(1,2):yspacing:bottomleft2(1,2));
    plot(x,y,'+b')

    grid_x=x;
    grid_y=y;

end

```

```

if gridselection==2
    %this is the grid with 2 rows, one top one bottom for
    boundary
    %displacements
    msgbox('click_top_left_corner_followed_by_bottom_right')

    [topleft2(1,1:2)]=ginput(1);
    hold on
    plot(topleft2(1,1),topleft2(1,2),'+g')

    [bottomright2(1,1:2)]=ginput(1);
    hold on
    plot(bottomright2(1,1),bottomright2(1,2),'+g')

    x1 = topleft2(1,1):xspacing:topleft2(1,1);
    x1(1,size(x1,2))=topleft2(1,1); %makes sure last point is
    exactly spaced

    %the following makes sure both rows have the same amount of
    points

    %if size(bottomleft2(1,1):xspacing2:bottomright2(1,1),2)~=size
    (x1,2)-1
    xspacing2=(bottomright2(1,1)-bottomleft2(1,1))/(size(x1,2)-1);
    xspacing2=abs(xspacing2);
    x2 = bottomleft2(1,1):xspacing2:bottomright2(1,1);
    %w2=size(x2,2);
    %x2(1,w2+1)=bottomright2(1,1); %makes sure last point is
    exactly spaced
    % else
    %     end
    x=[x1;x2];

    %Adjusts Y spacing so that the two rows are equal distances
    from the notch
    %it compares the locations of all 4 points first and picks
    the two that
    %are closest to begin with. Commented for testing other things
    than ipl
    %coupons

    %
    % diff(1)=dist1-dist2;
    % diff(2)=dist3-dist4;
    % diff(3)=dist1-dist4;
    % diff(4)=dist3-dist2;
    %
    % if diff(1)<diff(2)
    %     diff5=diff(1);
    %     dist=mean(dist1,dist2);
    % else

```

```

%      diff=diff(2);
%      dist=mean(dist3,dist4);
% end
% if diff(3)<diff(4)
%      diff6=diff(3)
%      dist=mean(dist1,dist4);
% else
%      diff
%      dist
% end
% if diff(4)<diff5

% diff(1)=dist1-dist2;
% diff(2)=dist3-dist4;
% if diff(1)<diff(2)
%      dist=min(dist1,dist2);
% else
%      dist=min(dist3,dist4);
% end

% dist(1)=notch-topright2(1,2);
% dist(2)=bottomleft2(1,2)-notch;
% dist(3)=notch-topleft2(1,2);
% dist(4)=bottomright2(1,2)-notch;
% dist=min(dist);
% y1=ones(1,size(x1,2))*(notch-dist);
% y2=ones(1,size(x2,2))*(notch+dist);
%uncomment above and the dist1,2,3,4 lines to have equally
    spaced grid
%points

%all y values are the same, uses the left side coordinates
y1=ones(1,size(x1,2))*topleft2(1,2);
y2=ones(1,size(x2,2))*bottomleft2(1,2);

y=[y1;y2];

plot(x,y,'+b')

grid_x=x;
grid_y=y;
%assembles notch points into a matrix and changes their
    coordinates to
%originate at the top left corner
notchpoints(1,1:2)=notchtop;
notchpoints(2,1:2)=notchbottom;
notchpoints(3,1:2)=notchcenter;

notchpoints(:,1)=notchpoints(:,1)-topleft2(1);

```

```

notchpoints(:,2)=notchpoints(:,2)-topleft2(2);

save notchpoints.dat notchpoints -ascii -tabs
end

% Do you want to keep the grid?
confirmselection = menu(sprintf('Do you want to use this grid?
    '),...
    'Yes','No');

if confirmselection==1
% Save settings and grid files in the image directory for
% visualization/plotting later
% save settings.dat xspacing yspacing
% xmin_new xmax_new ymin_new ymax_new -ascii -tabs
    save grid_x.dat x -ascii -tabs
    save grid_y.dat y -ascii -tabs
    hold off
    close all
    clear x* y* t* b* i* himage hpanel hpixinfo hfig hmagbox
    clear notchtop notchbottom
end

    if confirmselection==2
        close all
        hold off
%     imshow(im_grid,'true size');
%     IPL_grid_generator(FileNameBase);
end

```

automate_image.m

This function takes the grid points and the list of files and correlates all the points on all the images. It produces the output files validx.dat and validy.dat for further use.

```

function [validx,validy]=automate_image(grid_x,grid_y,
    filenamelist,validx,validy);

% Code to start actual image correlation
% Programmed by Chris and Rob
% Last revision: 12/26/06
% Modifications by JP at lines 107 and 119 6/11/2008 comments
    on those lines

% The automation function is the central function and
    processes all markers and
% images by the use of the matlab function cpcorr.m.

```

```

% Therefore the Current directory in matlab has to be the
% folder where
% automate_image.m finds the filenameslist.mat, grid_x.dat and
% grid_y.dat as well
% as the images specified in filenameslist.mat. Just type
% automate_image; and
% press ENTER at the command line of matlab.
% At first, automate_image.m will open the first image in the
% filenameslist.mat and
% plot the grid as green crosses on top. The next step will
% need some time since
% all markers in that image have to be processed for the first
% image. After correlating
% image one and two the new raster positions will be plotted
% as red crosses. On top
% of the image and the green crosses. The next dialog will ask
% you if you want to
% continue with this correlation or cancel. If you press
% continue, automate_image.m
% will process all images in the filenameslist.mat. The time it
% will take to process
% all images will be plotted on the figure but can easily be
% estimated by knowing the
% raster point processing speed (see processing speed).
% Depending on the number of images and markers you are
% tracking, this process
% can take between seconds and days. For 100 images and 200
% markers a decent
% computer should need 200 seconds. To get a better resolution
% you can always
% run jobs overnight (e.g. 6000 markers in 1000 images) with
% higher resolutions.
% Keep in mind that CORRSIZE which you changed in cpcorr.m
% will limit your
% resolution. If you chose to use the 15 pixel as suggested a
% marker distance of
% 30 pixel will lead to a full cover of the strain field.
% Choosing smaller marker
% distances will lead to an interpolation since two
% neighboring markers share
% pixels. Nevertheless a higher marker density can reduce the
% noise of the strain field.
% When all images are processed, automate_image will write the
% files validx.mat,
% validy.mat, validx.txt and validy.txt. The text files are
% meant to store the result in a
% format which can be accessed by other programs also in the
% future.

% exist('grid_x')
% exist('grid_y')
% exist('filenameslist')

```

```

% exist('validx')
% exist('validy')

% Load necessary files
if exist('grid_x')==0
    load('grid_x.dat')          % file with x position,
                                created by grid_generator.m
end
if exist('grid_y')==0
    load('grid_y.dat')          % file with y position,
                                created by grid_generator.m
end
if exist('filenamelist')==0
    load('filenamelist')        % file with the list of
                                filenames to be processed
end
resume=0;
if exist('validx')==1
    if exist('validy')==1
        resume=1;
        [Rasternum Imagenum]=size(validx);
    end
end

% Initialize variables
input_points_x=grid_x;
base_points_x=grid_x;

input_points_y=grid_y;
base_points_y=grid_y;

if resume==1
    input_points_x=validx(:,Imagenum);
    input_points_y=validy(:,Imagenum);
    inputpoints=1;
end

[row,col]=size(base_points_x);    % this will determine the
    number of rasterpoints we have to run through
[r,c]=size(filenamelist);        % this will
    determine the number of images we have to loop through

% Open new figure so previous ones (if open) are not
    overwritten
h=figure;
imshow(filenamelist(1,:))        % show the first image
title('Initial_Grid_For_Image_Correlation_(Note_green_crosses)')
    % put a title
hold on
plot(grid_x,grid_y,'g+')        % plot the grid onto the
    image

```

```

hold off

% Start image correlation using cpcorr.m
g = waitbar(0,sprintf('Processing images'));           %
    initialize the waitbar
set(g,'Position',[275,50,275,50])
                                % set the position of the
    waitbar [left bottom width height]
firstimage=1;

if resume==1
    firstimage=Imagenum+1
end

for i=firstimage:(r-1)          % run through all images

    tic                        % start the timer
    base = uint8(mean(double(imread(filenamelist(1,:))),3));
                                % read in the base image ( which is always
                                image number one. You might want to change that to
                                improve correlation results in case the light
                                conditions are changing during the experiment
    input = uint8(mean(double(imread(filenamelist((i+1),:))),
                                ,3));      % read in the image which has to be
                                correlated

    %-----
    %This is a modification by JP in June 2008. This is for
    the gradient
    %based method, it's a binomial low pass filter that
    removes high
    %frequency content and reduces the error substantially (at
    least on
    %artificially displaced images)
    %it's a 2-D binomial filter of order 2  $B^2$  from  $1/4*[1 \ 2 \ 1]$ 
    %this could be bad on the edges. . . probably won't matter
    though
    h=1/16*[1 2 1 ;2 4 2 ;1 2 1];
    base = filter2(h,base);
    input = filter2(h,input);
    %-----end-----

    input_points_for(:,1)=reshape(input_points_x,[],1);
                                % we reshape the input points to one row of
                                values since this is the shape cpcorr will accept
    input_points_for(:,2)=reshape(input_points_y,[],1);
    base_points_for(:,1)=reshape(base_points_x,[],1);
    base_points_for(:,2)=reshape(base_points_y,[],1);
    %=====JP modification choose your cpcorr
    =====

```

```

input_correl(:,:)=cpcorr_gradient(input_points_for,
    base_points_for, input, base);
%modified to try and implement gradient using
    cpcorr_gradient JP 6-08
%comment above for original, below for gradient based
input_correl(:,:)=cpcorr(input_points_for,
    base_points_for, input, base);           % here we go
    and give all the markers and images to process to
    cpcorr.m which is a function provided by the matlab
    image processing toolbox
%=====/jp modification=====
input_correl_x=input_correl(:,1);
                                           % the results we
    get from cpcorr for the x-direction
input_correl_y=input_correl(:,2);
                                           % the results we
    get from cpcorr for the y-direction

validx(:,i)=input_correl_x;
                                           %
    lets save the data
savelinex=input_correl_x';
dlmwrite('resultsimcorr_x.txt', savelinex , 'delimiter', '\
t', '-append');           % Here we save the result from
    each image; if you are desperately want to run this
    function with e.g. matlab 6.5 then you should comment
    this line out. If you do that the data will be saved at
    the end of the correlation step - good luck ;-))

validy(:,i)=input_correl_y;
saveliney=input_correl_y';
dlmwrite('resultsimcorr_y.txt', saveliney , 'delimiter', '\
t', '-append');

waitbar(i/(r-1))

    % update the waitbar

% Update base and input points for cpcorr.m
base_points_x=grid_x;
base_points_y=grid_y;
input_points_x=input_correl_x;
input_points_y=input_correl_y;

imshow(filenamelist(i+1,:))
    image
hold on
plot(grid_x,grid_y,'g+')
    plot start position of raster
plot(input_correl_x,input_correl_y,'r+')
    actual position of raster
hold off

```

```

drawnow
time(i)=toc;

                                % take
    time
estimatedtime=sum(time)/i*(r-1);    % estimate
    time to process
[h,m,s] = hms2mat(timedim(estimatedtime,'seconds','hms'));
title(['#_Im.:_', num2str((r-1)),';_Proc._Im._#:_',
    num2str((i)),';_#_Rasterp.:_',num2str(row*col), '_;_Est._
    Time_[h:m:s]_', num2str(h),':',num2str(m),':',num2str(s
    )])    % plot a title onto the image
drawnow

end

close(g)
close all

% save

save time.dat time -ascii -tabs
save validx.dat validx -ascii -tabs
save validy.dat validy -ascii -tabs

```

JP_boundary_displacements.m

This function calculates average boundary displacements for the sample from the results of the correlation. It outputs the X, Y, and R displacements of the top and bottom boundaries for each step into MATLAB and plots them.

```

%started by JP on 4/19/07 It requires that grid_X and grid_y
    have
%only 2 rows of points, one
%at the top of the desired gage section and one at the bottom.
    The
%IPL_strain_grid_generator.m program will work for this.

% Load necessary files
%if exist('grid_x')==0
    load('grid_x.dat')
    % file with x position, created by grid_generator.m
%end
%if exist('grid_y')==0
    load('grid_y.dat')
    % file with y position, created by grid_generator.m
%end
%if exist('filenamelist')==0
    load('filenamelist')
    % file with the list of filenames to be processed
%end

```

```

if exist('notchpoints')==0
    load ('notchpoints.dat')
end

resume=0;
%if exist('validx')==0
    load('validx.dat')
    %    if exist('validy')==0
        load('validy.dat')

    %    end
%end

[ypoints xpoints]=size(grid_x);
%number of grid points across the sample
[Rasternum Imagenum]=size(validx);
% rasternum number of points in validx file (total points on
    sample),
%imagenum is number of images

% show the first image
imshow(filenamelist(1,:), 'InitialMagnification', 50)
title('original_positions(green)'...
    ... displaced positions red and blue)...
    ...center displacements (yellow) linear curve fits (blue)')
hold on
plot(grid_x, grid_y, 'g+')

%the following finds center points of the top and bottom grids
    then plots
%them.
center=[1 grid_y(1,1); 1 grid_y(2,1)] ;
center(1,1)=(grid_x(1,1)+(grid_x(1,size(grid_x,2))-grid_x(1,1)
    )/2);
center(2,1)=(grid_x(2,1)+(grid_x(2,size(grid_x,2))-grid_x(2,1)
    )/2);
plot(center(1,1), center(1,2), 'y*', center(2,1), center(2,2), 'y*'
    )

%makes an empty matrix
fitdata = ones(Imagenum, 4);

%makes matrixes of x distances from center
working_x=grid_x;
working_y=grid_y;
working_x(1,:) = working_x(1,:)-center(1,1);
working_x(2,:) = working_x(2,:)-center(2,1);
%uses width of sample (from dialog)and width in pixels to
    convert to
%inches for the displacement output. It does this for the top
    and

```

```

%bottom separately, the two can be compared to check alignment
/distortion
prompt = {'Enter sample width at boundary in desired
displacement units'};
dlg_title = 'Width calibration';
num_lines = 1;
def = {'1.0'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
width=answer{1};
width=str2double(width);

topscale=(grid_x(1,xpoints)-grid_x(1,1))/width;
bottomscale=(grid_x(2,xpoints)-grid_x(2,1))/width;
% topscale=1;
% bottomscales=1;
%above to give pixel displacements
for ijk = 1:Imagenum

%this will plot all the displaced positions on the same image.
Only works
%if there are only 2 lines in the y dimension in gridx
plot(validx(2:2:Rasternum,ijk),validy(2:2:Rasternum,ijk),...
'r+',validx(1:2:Rasternum,ijk),validy(1:2:Rasternum,ijk),'
b+')

%fits curves to lines
fitdata(ijk,1:2) = polyfit(validx(2:2:Rasternum,ijk)...
,validy(2:2:Rasternum,ijk),1); %fits curves to bottom
lines
fitdata(ijk,3:4) = polyfit(validx(1:2:Rasternum,ijk)...
,validy(1:2:Rasternum,ijk),1); %fits curves to top
displacements

%plots the curves on the image for visual confirmation that
they are
%reasonable
xvals=grid_x(1,1):10:grid_x(1,xpoints);
yvals= fitdata(ijk,4)+fitdata(ijk,3)*xvals;
plot(xvals,yvals,'c')
xvals2=grid_x(2,1):10:grid_x(2,xpoints);
yvals2= fitdata(ijk,2)+fitdata(ijk,1)*xvals2;
plot(xvals2,yvals2,'c')

%convert slope to angle. useually they are amall enough that
slope and
%angle are equal values. adds them as columns 5 and 6
displacement(ijk,5)=atan(fitdata(ijk,1)); %top row
displacement(ijk,6)=atan(fitdata(ijk,3)); %bottom row

%rotates working_x by theta,
%finds average distances of points on both rows.
for lmn=1:xpoints

```

```

% working_x(1,lmn)=working_x(1,lmn)*cos(theta(ijk,1));
% working_y(1,lmn)=working_x(1,lmn)*sin(theta(ijk,1));
working_x(1,lmn)=working_x(1,lmn)*cos(displacement(ijk,5));
working_y(1,lmn)=working_x(1,lmn)*sin(displacement(ijk,5));

xdist(lmn,1)=validx(lmn*2-1,ijk)-working_x(1,lmn);
ydist(lmn,1)=validy(lmn*2-1,ijk)-working_y(1,lmn);

% working_x(2,lmn)=working_x(2,lmn)*cos(theta(ijk,2));
% working_y(2,lmn)=working_x(2,lmn)*sin(theta(ijk,2));
working_x(2,lmn)=working_x(2,lmn)*cos(displacement(ijk,6))
;
working_y(2,lmn)=working_x(2,lmn)*sin(displacement(ijk,6))
;

x2dist(lmn,1)=validx(lmn*2,ijk)-working_x(2,lmn);
y2dist(lmn,1)=validy(lmn*2,ijk)-working_y(2,lmn);
end

% takes an average of the distances and saves displacement
% info as x,y
% coordinates in inches.
displacement(ijk,1)=(mean(xdist)-center(1,1))/topscale;
displacement(ijk,2)=(mean(ydist)-center(1,2))/topscale;
displacement(ijk,3)=(mean(x2dist)-center(2,1))/bottomscale
;
displacement(ijk,4)=(mean(y2dist)-center(2,2))/bottomscale
;

end
%saves gage length data
displacement(ijk+1,1)=(grid_y(2,1)-grid_y(1,1))/(topscale);
displacement(ijk+1,2)=(grid_y(2,xpoints)-grid_y(1,xpoints))/(
bottomscale);
%saves notch location
displacement(ijk+1,3:4)=notchpoints(3,1:2)/topscale;

plot(displacement(:,1)*topscale+center(1,1),...
displacement(:,2)*topscale+center(1,2),'y+');
plot(displacement(:,3)*bottomscale+center(2,1),...
displacement(:,4)*bottomscale+center(2,2),'y+');
%plot(working_x,working_y)
%this plots the lines that have been fit for inspection
xvals=grid_x(1,1):10:grid_x(1,xpoints);
yvals= fitdata(5,4)+fitdata(5,3)*xvals;
plot(xvals,yvals,'c')

save boundarydisplacements.dat displacement -ascii -tabs

hold off
%close all

```

```
clear x* y* i* j* a* b* v* width f* t*
```

JP_inplanestrains.m

This function uses central difference displacement gradient approximations to generate $\epsilon_y y$, $\epsilon_x x$ and ϵ_{xy} strain data from an evenly spaced strain grid. It will work with a grid that has different x and y grid point spacing too.

```
%calculates a 2-d strain field from a validx and validy files
    gathering
%information from grid_x and grid_y. Modified in june 08 by
    JP as a new
%version that uses central difference approximations for the
    partial
%derivatives rather than the complicated method used before.
    It seems to
%be working pretty well
```

```
% Load necessary files
load('grid_x.dat') % file with x position, created by
    grid_generator.m
load('grid_y.dat') % file with y position, created by
    grid_generator.m
load('filenamelist')% file with the list of filenames to be
    processed
load('validx.dat')
load('validy.dat')
```

```
resume=0;
%===requires evenly spaced grid points===
```

```
%distances between X and Y points as defined in grid
    generation
```

```
xdist = abs(grid_x(1,2)-grid_x(1,1));
ydist = abs(grid_y(2,1)-grid_y(1,1));
%Number of correlation points in x and y axis'
ypoints = size(grid_x,1);
xpoints = size(grid_x,2);
%number of images
steps= size(validx,2);
```

```
%makes a 3d version of the displacements, validx and validy
```

```
for i=1:steps
    for j=1:xpoints
        validx3d(1:ypoints,j,i)...
            =validx(1+(j-1)*ypoints:1+(j-1)*ypoints+ypoints-1,
                i);
        validy3d(1:ypoints,j,i)...
            =validy(1+(j-1)*ypoints:1+(j-1)*ypoints+ypoints-1,
                i);
    end
end
```

```

end
%subtracts original positions out of validx3d and validy3d to
    leave only
%the x and y displacements, respectively

%binomial low pass filter for smoothing things out a little
%2-D binomial filter of order 2  $B^2$  from  $1/4*[1 \ 2 \ 1]$ 
    h=1/16*[1 2 1 ;2 4 2 ;1 2 1];
%this seems to help significantly

for i=1:steps
    xdisp(:,:,i)=validx3d(:,:,i)-grid_x;
    ydisp(:,:,i)=validy3d(:,:,i)-grid_y;
    xdisp(:,:,i) = filter2(h,xdisp(:,:,i));
    ydisp(:,:,i) = filter2(h,ydisp(:,:,i));

end
%convolution filter/masks for approximating partial
    derivatives,
%it's a central difference approximation of error  $O($ 
    gridspace $)^4$ 
    filtx=1/xdist*1/12*[-1 8 0 -8 1];
    %transposed for y direction.
    filty=(1/ydist*1/12*[-1 8 0 -8 1])';

%preallocate, two points on each edge are lost because full
    overlap of the
%filter is needed

%X direction partials. 'valid' returns elements with full
    convolution
%overlap only. using the lagrange strain tensor, the partial
    of the x
%displacement in x is the x strain, likewise for y. grid_x and
    grid_y are
%subtracted so only the displacements remain.
for i=1:steps
    xstrain3d(:,:,i)=conv2(xdisp(:,:,i),filtx,'valid');
%Y direction partials.
    ystrain3d(:,:,i)=conv2(ydisp(:,:,i),filty,'valid');
%shear strain: get the shear angles by using the same finite
    difference
%approximation only do the y direction filter on the x
    displacements and
%the x direction filter on the y displacements. These are
    small so the
%small angle approximation of the tangent of these angles is
    equal to the
%angles is equal to the strains.
    xystrain3d(:,:,i)=conv2(xdisp(:,:,i),filty,'valid');
    yxstrain3d(:,:,i)=conv2(ydisp(:,:,i),filtx,'valid');
end

```

```

%Median filter to smooth things out.  symmetric pads the edges
    with the
%value on the other side rather than zero to avoid making the
    corners
%filter to zero.  According to testing this filter actually
    makes the data
%less accurate, but sometimes it makes it look better for
    plotting
for k=1:steps
ystrain3dsmooth(:,:,k)=medfilt2(ystrain3d(:,:,k),'symmetric');
xstrain3dsmooth(:,:,k)=medfilt2(xstrain3d(:,:,k),'symmetric');
xystrain3dsmooth(:,:,k)=medfilt2(xystrain3d(:,:,k),'symmetric'
    );
yxstrain3dsmooth(:,:,k)=medfilt2(yxstrain3d(:,:,k),'symmetric'
    );
end

%trim them down to the same size.

xstrain3d=xstrain3d(4:ypoints-3,2:xpoints-5,:);
ystrain3d=ystrain3d(2:ypoints-5,4:xpoints-3,:);
yxstrain3d=yxstrain3d(4:ypoints-3,2:xpoints-5,:);
xystrain3d=xystrain3d(2:ypoints-5,4:xpoints-3,:);

xstrain3dsmooth=xstrain3dsmooth(4:ypoints-3,2:xpoints-5,:);
ystrain3dsmooth=ystrain3dsmooth(2:ypoints-5,4:xpoints-3,:);
yxstrain3dsmooth=yxstrain3dsmooth(4:ypoints-3,2:xpoints-5,:);
xystrain3dsmooth=xystrain3dsmooth(2:ypoints-5,4:xpoints-3,:);

shearstrain3d=1/2*(xystrain3d+yxstrain3d);
shearstrain3dsmooth=1/2*(xystrain3dsmooth+yxstrain3dsmooth);

%make some trimmed displacement grids for the sake of plotting
    the strains
svalidx3d=validx3d(4:ypoints-3,4:xpoints-3,:);
svalidy3d=validy3d(4:ypoints-3,4:xpoints-3,:);

%remove bad data - things over max strtain defined below

%%%!!moved into strain visualizer program !!
%-----
% maxstrain=1;
% %-----
% [x,y,z]=size(xstrain3d);
% for k=1:z;
%     for j=1:y;
%         for i=1:x;
%             if abs(xstrain3d(i,j,k))>maxstrain
%                 xstrain3d(i,j,k)=0;
%             else
%                 end
%         end
%     end
% end

```

```

%           if abs(ystrain3d(i,j,k))>maxstrain
%               ystrain3d(i,j,k)=0;
%           else
%               end
%           if abs(xystrain3d(i,j,k))>maxstrain
%               xystrain3d(i,j,k)=0;
%           else
%               end
%           end
%       end
%   end

%optional second median filters the strain data to reduce
%   noise
for k=1:steps
ystrain3dsmooth(:,:,k)=medfilt2(ystrain3d(:,:,k),'symmetric');
xstrain3dsmooth(:,:,k)=medfilt2(xstrain3d(:,:,k),'symmetric');
xystrain3dsmooth(:,:,k)=medfilt2(xystrain3d(:,:,k),'symmetric'
);
shearstrain3dsmooth(:,:,k)=medfilt2(shearstrain3d(:,:,k),'
symmetric');
end

clear resume maxstrain j i j k x y z ypoints xpoints ex ey h
    filtx filty
%housecleaning
%xdist ydist xdisp ydisp
%clear grid_y grid_x points resume xpoints ypoints i j k x y
z
%
%the code below calculates average x and y strains for uni-
axial purposes
for i=1:steps
    shearstrain(i)=mean(mean(shearstrain3d(:,:,i)));
    ystrain(i)=mean(mean(ystrain3d(:,:,i)));
    xstrain(i)=mean(mean(xstrain3d(:,:,i)));
end
    shearstrain=shearstrain';
    ystrain=ystrain';
    xstrain=xstrain';

    j=0 ;
for i=2:2:26
    j=j+1;
    top(j,:)=validy(i,:);
    bottom(j,:)=validy(i-1,:);
end

```

JP_strain_visualizer_centraldiff.m

The script JP_strain_visualizer.m is used to look at the strain information overlaid on the deformed images and export them for publication or as countour maps to better visualize the data.

%plots strain over the image to help explain what's going on.

Created by

%JP 10-2007

```
% load('filenamelist')
straintypes=['ystrain3dsmooth_';'xstrain3dsmooth';...
'yxstrain3dsmooth';'ystrain3d_';...
';'xstrain3d_';'yxstrain3d_'];
```

```
[s,v] = listdlg('PromptString','Select a step',...
'SelectionMode','single',...
'ListString',filenamelist);
[t,u] = listdlg('PromptString','Select strain',...
'SelectionMode','single',...
'ListString',straintypes) ;
```

```
image=filenamelist(s,:);
image=imread (image);
if t==1
    str=ystrain3dsmooth;
    titl='Y-Strain_median_filtered';
end
if t==2
    str=xstrain3dsmooth;
    titl='X-Strain_median_filtered';
end
if t==3
    str=shearstrain3dsmooth;
    titl='XY-Strain_median_filtered';
end
if t==4
    str=ystrain3d;
    titl='Y-Strain';
end
if t==5
    str=xstrain3d;
    titl='X-Strain';
end
if t==6
    str=shearstrain3d;
    titl='XY-Strain';
end
```

```
smin=-max(max(max(-str(:,:s-1))));
smax=max(max(max(str(:,:s-1))));
```

```

%the layer that will be plotted is seperated, masked then 0's
    are changed
%to NaN

plot=str(:,:,s-1);
%all the terms that are 0 in mask will become 0 in plot
plot=plot.*mask;
%scaled up so it will plot correctly

plot=plot*10000;
[x y]=size(plot);
for i=1:x;
    for j=1:y;
        if plot(i,j)==0;
            plot(i,j)=NaN;
        end
    end
%for step 50 y strains use 275 and -350
%for xy strains use 300 and -100
        if plot(i,j)>=500;
            plot(i,j)=NaN;
        end
        if plot(i,j)<=-200;
            plot(i,j)=NaN;
        end
    end
end
    plot2=plot/10000;

subimage(image);
hold on
%annoyingly, I can't get it to plot using the scale of the
    actual strain
%data so I'm multiplying it by a 1000 so it's easy to convert
%used to be surf not contour below
h=surf(svalidx3d(:,:,s-1),svalidy3d(:,:,s-1),plot);
%contour labels
%f=clabel(C,h);
%set(f,'BackgroundColor',[1 1 1]);
%\countour labels
title([titl,'_on_step_number_',num2str(s)]);
set(h,'line','none')
set(h,'facealpha',.75);
set(h,'marker','+', 'markersize',3);
set(h,'cdatamapping','scaled');
%shading interp;
%doesn't seem to set the z axis scale correctly, and it messes
    other stuff
%up too on top of it. .
%axis([0 1157 0 1737 smin smax]);
%view ([0,90]);

hold off

```

```
clear dlg_title prompt t u v straintypes
```

Other Code Used

The script JP_boundary_grid_t0_strain_grid.m converts a boundary displacement grid to a strain grid maintaining the corner locations. This function was made to help with using DIC data in the formulation of a DED function.

```
load ('grid_x.dat');
load('grid_y.dat');
load('boundarydisplacements.dat');

load('filenamelist.mat')
if exist('notchpoints')==0
    load ('notchpoints.dat')
end

hfig = figure('Menubar', 'none',...
              'Name','How does this look?',...
              'NumberTitle','off',...
              'IntegerHandle','off');
im_grid = imread(filenamelist(1,:));
himage = imshow(im_grid);
hold on

%finds last point across sample to extract width, finds the
%row of the
%boundary displacements file with GL and notch information in
it
maxX=size(grid_x,2);
glrow=size(boundarydisplacements,1);

averagewidth=((grid_x(1,maxX)-grid_x(1,1))+grid_x(2,maxX)-
grid_x(2,1))/2;
% notchpoint(1,1)=boundarydisplacements(glrow,3);
% notchpoint(1,2)=boundarydisplacements(glrow,4);
topleft2(1,1)=grid_x(1,1);
topright2(1,1)=grid_x(1,maxX);
bottomleft2(1,1)=grid_x(2,1);
bottomright2(1,1)=grid_x(2,maxX);

topleft2(1,2)=grid_y(1,1);
topright2(1,2)=grid_y(1,maxX);
bottomleft2(1,2)=grid_y(2,1);
bottomright2(1,2)=grid_y(2,maxX);

% grid resolution. Strainfield will use this as an initial
value but
% change it so that the far edges will be covered.
```

```

xspacing = 15;
yspacing = 15;

%this is the regular spaced grid for strainfield use
%distances. Starts by making topleft and topright
%points from the othertwo that were originally selected

xdist=topright2(1,1)-topleft2(1,1);
ydist=bottomright2(1,2)-topright2(1,2);

%determines number of points that will fit
xpoints=floor(xdist/xspacing);
ypoints=floor(ydist/yspacing);
%redefines spacings so that the grid will fully cover the
    sample i.e.
%bottom and right edges will have points
xspacing = xdist/xpoints;
yspacing= ydist/ypoints;
%creates an evenly spaced grid...
%(with different x and y spacing however)
[x,y] = meshgrid(topleft2(1,1):...
    xspacing:topright2(1,1),topleft2(1,2)...
    :yspacing:bottomright2(1,2));
plot(x,y,'+b')

grid_x=x;
grid_y=y;
save grid_x.dat x -ascii -tabs
save grid_y.dat y -ascii -tabs

clear maxX b* x* y* n* t*

```

The script JP_fake_displacements.m was modified and used to make the artificially displaced images used for testing purposes. Images were created using MATLAB's affine transformations and also frequency domain shifted images were created using the script shiftimage.m which was provided through the MATLAB Central File Exchange by Edward Brian Welch, who can be contacted at edwardbrianwelch@yahoo.com .

```

%
% img_shifted = ShiftImage(img, frac_deltax, frac_deltay)
%
% Shifts the image IMG using the Fourier shift theorem by the
% Field-of-view (FOV) fractions FRAC_DELTAX and FRAC_DELTAY.
% The fraction is relative to the size of the image such that
% a value of 1.0 shifts the image back to where it started.
% Shifts outside this range will wrap around to the
% corresponding shift within the range.
%
% The shift is performed using Fourier sinc interpolation, i.
e.
% the Fourier shift theorem, via simple phase adjustments in

```

```

% the frequency domain. This may lead to Gibbs phenomena.
%
% Dimensions need not be powers of 2 and will not necessarily
% be padded to a power of 2. The Matlab FFT functions are
% capable of non-power of 2 transforms.
%
% Example:
%     load mri;
%     img = double(squeeze(D(:,:,1,13)));
%     figure;
%     dx = [.1 .5 .8 1.2];
%     dy = [.2 .7 1.0 2.5];
%     for n=1:4,
%         subplot(2,2,n);
%         imagesc( ShiftImage(img,dx(n),dy(n)) );
%         axis image; colormap(gray);
%     end
%
%
%
% 01/25/99 Implemented by Edward Brian Welch,
%     edwardbrianwelch@yahoo.com
%
function [Image] = shiftimage(Image, deltax, deltay)

% NOTE:
% This function is partially vectorized (only single loops,
% no double loops) to gain some benefit in speed. The
% tradeoff
% is that more memory is required to hold the large matrices.
% It would be possible to completely vectorize certain parts
% of the mfile, but that would require large amounts of memory
% and would also make the code less readable.

[oxdim oydim] = size(Image);

if mod(oxdim,2)==1 & mod(oydim,2)==1,
    tmp = zeros(oxdim+1,oydim+1);
    tmp(1:oxdim,1:oydim) = Image;
    Image = tmp;
    xdim = oxdim+1;
    ydim = oydim+1;
elseif mod(oxdim,2)==1 & mod(oydim,2)==0,
    tmp = zeros(oxdim+1,oydim);
    tmp(1:oxdim,1:oydim) = Image;
    Image = tmp;
    xdim = oxdim+1;
    ydim = oydim;
elseif mod(oxdim,2)==0 & mod(oydim,2)==1,
    tmp = zeros(oxdim+1,oydim);
    tmp(1:oxdim,1:oydim) = Image;
    Image = tmp;

```

```

    xdim = oxdim;
    ydim = oydim+1;
else
    xdim = oxdim;
    ydim = oydim;
end

% Determine type of image to return
if isreal(Image),
    real_flag = 1;
else
    real_flag = 0;
end

% Put deltas into the range [-1,1]
while deltax<0,
    deltax = deltax + 2;
end

while deltax>2,
    deltax = deltax - 2;
end

while deltax>2,
    deltax = deltax - 2;
end

while deltax>2,
    deltax = deltax - 2;
end

while deltax>2,
    deltax = deltax - 2;
end

% Calculate image's center coordinates
xno = (xdim-1)/2;
yno = (ydim-1)/2;

% Forward FFT image rows
Image = fft(Image, xdim, 1);

% Prepare to use the Fourier Shift Theorem
%  $f(x-deltax) \Leftrightarrow \exp(-j*2*pi*deltax*k/N)*F(k)$ 

% Initialize constant part of the exponent
% expression.
const1 = (-2.0*pi/(xdim)) * deltax * xdim;

% Calculate k values (Nyquist is at x=xno)
k_array = zeros(xdim,1);

for x=1:xdim,
    if (x-1)<=xno,
        k_array(x) = (x-1);
    else

```

```

        k_array(x) = (x-1-xdim);
    end
end

for y=1:ydim,

    % Calculate the angles
    angle_array = cons1*k_array;

    % Rotate the complex numbers by those angles
    sin_ang = sin(angle_array);
    cos_ang = cos(angle_array);
    newr = real(Image(:,y)).*cos_ang - imag(Image(:,y)).*
        sin_ang;
    newi = real(Image(:,y)).*sin_ang + imag(Image(:,y)).*
        cos_ang;
    Image(:,y) = newr + newi*i;

end

%-----
% SECOND SHIFT
%-----

Image = ifft(Image, xdim, 1);
Image = fft(Image, ydim, 2);

% Initialize constant part of the exponent
% expression.
cons1 = (-2.0*pi/(ydim)) * deltay * ydim;

% Calculate k values (Nyquist is at y=yno)
k_array = zeros(ydim,1);

for y=1:ydim,
    if (y-1)<=yno,
        k_array(y) = (y-1);
    else
        k_array(y) = (y-1-ydim);
    end
end

for x=1:xdim,

    % Calculate the angles
    angle_array = cons1*k_array;

    % Rotate the complex numbers by those angles
    sin_ang = sin(angle_array);
    cos_ang = cos(angle_array);
    newr = real(Image(x,:)).*cos_ang' - imag(Image(x,:)).*
        sin_ang';

```

```

    newi = real(Image(x,:)).*sin_ang' + imag(Image(x,:)).*
        cos_ang';
    Image(x,:) = newr + newi*i;

end

```

```

Image = ifft(Image, ydim, 2);

```

```

% Return a Real image if original Image was Real
if real_flag==1,
    Image = real(Image);
end

```

```

% Return an Image of the original size (in case it was odd)
Image = Image(1:oxdim,1:oydim);

```

The MATLAB correlation file CPCORR.m was heavily modified for the gradient based method of DIC. The modified version is kept in a subfolder from the working directory with the name PRIVATE, which is a way to make MATLAB search there first when the function is called. It's code is as follows.

```

function xyinput = cpcorr(varargin)
%CPCORR Tune control point locations using cross-correlation.
% INPUT_POINTS = CPCORR(INPUT_POINTS_IN,BASE_POINTS_IN,INPUT
% ,BASE) uses
% normalized cross-correlation to adjust each pair of
% control points
% specified in INPUT_POINTS_IN and BASE_POINTS_IN.
%
% INPUT_POINTS_IN must be an M-by-2 double matrix containing
% the
% coordinates of control points in the input image.
% BASE_POINTS_IN is
% an M-by-2 double matrix containing the coordinates of
% control points
% in the base image.
%
% CPCORR returns the adjusted control points in INPUT_POINTS
% , a double
% matrix the same size as INPUT_POINTS_IN. If CPCORR cannot
% correlate a
% pairs of control points, INPUT_POINTS will contain the
% same coordinates
% as INPUT_POINTS_IN for that pair.
%
% CPCORR will only move the position of a control point by
% up to 4
% pixels. Adjusted coordinates are accurate up to one tenth
% of a
% pixel. CPCORR is designed to get subpixel accuracy from
% the image
% content and coarse control point selection.
%

```

```

% Note that the INPUT and BASE images must have the same
% scale for
% CPCORR to be effective.
%
% CPCORR cannot adjust a point if any of the following occur
% :
%   - points are too near the edge of either image
%   - regions of images around points contain Inf or NaN
%   - region around a point in input image has zero standard
%     deviation
%   - regions of images around points are poorly correlated
%
% Class Support
% -----
% The images can be numeric and must contain finite values.
% The input
% control point pairs are double.
%
% Example
% -----
% This example uses CPCORR to fine-tune control points
% selected in an
% image. Note the difference in the values of the
% INPUT_POINTS matrix
% and the INPUT_POINTS_ADJ matrix.
%
%     input = imread('onion.png');
%     base = imread('peppers.png');
%     input_points = [127 93; 74 59];
%     base_points = [323 195; 269 161];
%     input_points_adj = cpcorr(input_points,base_points,...
%                               input(:,:,1),base(:,:,1))
%
% See also CP2TFORM, CPSELECT, NORMXCORR2, IMTRANSFORM.
%
% Copyright 1993-2004 The MathWorks, Inc.
% $Revision: 1.16.4.4 $ $Date: 2004/10/20 17:54:13 $
%
% Input-output specs
% -----
% INPUT_POINTS_IN: M-by-2 double matrix
%                 INPUT_POINTS_IN(:)>=0.5
%                 INPUT_POINTS_IN(:,1)<=size(INPUT,2)+0.5
%                 INPUT_POINTS_IN(:,2)<=size(INPUT,1)+0.5
%
% BASE_POINTS_IN: M-by-2 double matrix
%                BASE_POINTS_IN(:)>=0.5
%                BASE_POINTS_IN(:,1)<=size(BASE,2)+0.5
%                BASE_POINTS_IN(:,2)<=size(BASE,1)+0.5
%
% INPUT: 2-D, real, full matrix
%         logical, uint8, uint16, or double
%         must be finite (no NaNs, no Infs inside

```

```

%           regions being correlated)
%
%   BASE:    2-D, real, full matrix
%           logical, uint8, uint16, or double
%           must be finite (no NaNs, no Infs inside regions
%           %being correlated)

[xyinput_in,xybase_in,input,base] = ParseInputs(varargin{:});

%correlation size.  This seems to work well at 10
CORRSIZE = 10;

% get all rectangle coordinates
rects_input = calc_rects(xyinput_in,CORRSIZE,input);
rects_base = calc_rects(xybase_in,2*CORRSIZE,base);

ncp = size(xyinput_in,1);

xyinput = xyinput_in; % initialize adjusted control points
matrix

for icp = 1:ncp

    if isequal(rects_input(icp,3:4),[0 0]) || ...
        isequal(rects_base(icp,3:4),[0 0])
        % near edge, unable to adjust
        continue
    end

    sub_input = imcrop(input,rects_input(icp,:));
    sub_base = imcrop(base,rects_base(icp,:));

%scale interpolates image size up, may increase accuracy but
%definitely increases processing time. . . . a lot -JP '07
%addendum '08, 2x interpolation seems to help, tried up to 8x
%with no
%additional improvement - see interpolation study spreadsheet
% scale=1;
%sub_input=imresize(sub_input,scale,'bicubic');
%sub_base=imresize(sub_base,scale,'bicubic');

    inputsize = size(sub_input);

% make sure finite
if any(~isfinite(sub_input(:))) || any(~isfinite(sub_base
(:)))
    % NaN or Inf, unable to adjust
    continue
end

% check that template rectangle sub_input has nonzero std

```

```

if std(sub_input(:))==0
    % zero standard deviation of template image, unable to
    adjust
    continue
end

norm_cross_corr = normxcorr2(sub_input,sub_base);

% Setting this to false makes findpeak return only pixel
accurate
%peak data so the gradient method can be used to get subpixel
data
subpixel = false;

[xpeak, ypeak, amplitude] = findpeak(norm_cross_corr,
    subpixel);

% eliminate any poor correlations
THRESHOLD = 0.5;
if (amplitude < THRESHOLD)
    % low correlation, unable to adjust
    continue
end

% offset found by cross correlation
%modified to reflect scaling of image done above JP 9-2007
corr_offset = [ ((xpeak-inputsize(2))/scale-CORRSIZE)...
    ((ypeak-inputsize(1))/scale-CORRSIZE) ];

=====
$$$$$$$$$$$$$$$ New Code Here jp-6-08 $$$$$$$$$$$$

%limits pixel displacements to 10 so there is room for
gradient method,
%i.e. sub image will always fully overlap base image
if abs(corr_offset(1))>=CORRSIZE
corr_offset(1)=0;
end
if abs(corr_offset(2))>CORRSIZE
corr_offset(2)=0;
end

%convolution filter/masks for approximating
%partial derivatives, it's a central
%difference approximation of order  $O(h)^4$  with  $h=1$ 

```

```

filtx=[1/12 -8/12 0 8/12 -1/12];
%transpose for y direction
filty=filtx';

%convolute sub image with masks to get partial derivative
approximations
for i=1:size(sub_input,2)
    Gx(i,:)=conv(filtx,sub_input(i,:));
end
%trim extra off - 4 from each end
Gx=Gx(3:size(Gx,1)-2,5:size(Gx,2)-4);

%compute the y partial the same way
for i=1:size(sub_input,1)
    Gy(:,i)=conv(filty,sub_input(:,i));
end
Gy=Gy(5:size(Gy,1)-4,3:size(Gy,2)-2);

%compute sums for least squares solution to truncated taylor
expansion
Gx2=sum(sum(Gx.*Gx));
Gy2=sum(sum(Gy.*Gy));
Gxy=sum(sum(Gx.*Gy));

%pixel level shift of image from center? I hope. . .
xshift=CORRSIZE+corr_offset(1,1);
yshift=CORRSIZE+corr_offset(1,2);

%base image portion under the sub image including the pixel
%displacements
sub_base_shift=sub_base(yshift+3:yshift+inputsize(1,1)
-2,...
    xshift+3:xshift+inputsize(1,2)-2);

sub_input_shift=sub_input(3:size(sub_input,1)-2,...
    3:size(sub_input,2)-2);

Gxt=sum(sum((sub_base_shift-sub_input_shift).*Gx));
Gyt=sum(sum((sub_base_shift-sub_input_shift).*Gy));

mat=[Gx2 Gxy;Gxy Gy2];
%subpixel displacements?
disp=inv(mat)*[Gxt;Gyt];

corr_offset=corr_offset+disp';
clear Gx Gy Gx Gxt Gxy Gyt xshift yshift

%+++++ /new stuff +++++

% eliminate any big changes in control points
ind = find(abs(corr_offset) > (CORRSIZE-1));

```

```

if ~isempty(ind)
    % peak of norxcorr2 not well constrained, unable to
    adjust
    continue
end

input_fractional_offset = xyinput(icp,:) - round(xyinput(
    icp,:));
base_fractional_offset = xybase_in(icp,:) - round(
    xybase_in(icp,:));

% adjust control point
xyinput(icp,:) = xyinput(icp,:) - input_fractional_offset
...
- corr_offset + base_fractional_offset;

end

%-----
%
function rect = calc_rects(xy, halfwidth, img)

% Calculate rectangles so imcrop will return image
% with xy coordinate inside center pixel

default_width = 2*halfwidth;
default_height = default_width;

% xy specifies center of rectangle, need upper left
upperleft = round(xy) - halfwidth;

% need to modify for pixels near edge of images
upper = upperleft(:,2);
left = upperleft(:,1);
lower = upper + default_height;
right = left + default_width;
width = default_width * ones(size(upper));
height = default_height * ones(size(upper));

% check edges for coordinates outside image
[upper, height] = adjust_lo_edge(upper, 1, height);
[dum, height] = adjust_hi_edge(lower, size(img, 1), height);
[left, width] = adjust_lo_edge(left, 1, width);
[dum, width] = adjust_hi_edge(right, size(img, 2), width);

% set width and height to zero when less than default size
iw = find(width < default_width);
ih = find(height < default_height);
idx = unique([iw; ih]);
width(idx) = 0;
height(idx) = 0;

rect = [left upper width height];

```

```

%-----
%
function [coordinates, breadth] = ...
    adjust_lo_edge(coordinates, edge, breadth)

indx = find( coordinates < edge );
if ~isempty(indx)
    breadth(indx) = breadth(indx) - abs(coordinates(indx) - edge
    );
    coordinates(indx) = edge;
end

%-----
%
function [coordinates, breadth] = ...
    adjust_hi_edge(coordinates, edge, breadth)

indx = find( coordinates > edge );
if ~isempty(indx)
    breadth(indx) = breadth(indx) - abs(coordinates(indx) - edge
    );
    coordinates(indx) = edge;
end

%-----
%
function [xyinput_in, xybase_in, input, base] = ParseInputs(
    varargin)

iptchecknargin(4, 4, nargin, mfilename);

xyinput_in = varargin{1};
xybase_in = varargin{2};
if size(xyinput_in, 2) ~= 2 || size(xybase_in, 2) ~= 2
    msg = sprintf(...
        'In function %s, control point matrices must be M-by-2.',
        mfilename);
    eid = sprintf('Images:%s:cpMatrixMustBeMby2', mfilename);
    error(eid, msg);
end

if size(xyinput_in, 1) ~= size(xybase_in, 1)
    msg = sprintf(...
        ('INPUT and BASE images need same number of control points.'
        , mfilename);
    eid = sprintf('Images:%s:needSameNumOfControlPoints',
        mfilename);
    error(eid, msg);
end

input = varargin{3};
base = varargin{4};

```

```

if ndims(input) ~= 2 || ndims(base) ~= 2
    msg = sprintf...
    ('In function %s, Images must be intensity images.',
     mfilename);
    eid = sprintf('Images:%s:intensityImagesReq', mfilename);
    error(eid, msg);
end

input = double(input);
base = double(base);

if any(xyinput_in(:)<0.5) || any(xyinput_in(:,1)>size(input,2)
    +0.5) || ...
    any(xyinput_in(:,2)>size(input,1)+0.5) || ...
    any(xybase_in(:)<0.5) || any(xybase_in(:,1)>size(base,2)
    +0.5) || ...
    any(xybase_in(:,2)>size(base,1)+0.5)
    msg = sprintf...
    ('In function %s, Control Points must be in pixel coordinates.',
     mfilename);
    eid = sprintf('Images:%s:cpPointsMustBeInPixCoord',
     mfilename);
    error(eid, msg);
end

```

The MATLAB peak finding code `findpeak.m` was also modified to get better resolution, although it is not used for the gradient method which only calls for pixel level data from `findpeak`. The modified code is below.

```

function [xpeak, ypeak, max_f] = findpeak(f, subpixel)
%FINDPEAK Find extremum of matrix.
% [XPEAK, YPEAK, MAX_F] = FINDPEAK(F, SUBPIXEL) finds the
    extremum of F,
% MAX_F, and its location (XPEAK, YPEAK). F is a matrix.
%MAX_F is the maximum
%absolute value of F, or an estimate of the extremum if a
    subpixel
%extremum is requested.
%
%SUBPIXEL is a boolean that controls if FINDPEAK attempts to
    estimate the
%extremum location to subpixel precision. If SUBPIXEL is false
    , FINDPEAK
%returns the coordinates of the maximum absolute value of F
    and MAX_F is
%max(abs(F(:))). If SUBPIXEL is true, FINDPEAK fits a 2nd
    order
%polynomial to the 9 points surrounding the maximum absolute
    value of
%F. In this case, MAX_F is the absolute value of the
    polynomial evaluated
%at its extremum.
%

```

```

%Note: Even if SUBPIXEL is true, there are some cases that
      result
%in FINDPEAK returning the coordinates of the maximum absolute
      value
%of F:
%* When the maximum absolute value of F is on the edge of
      matrix F.
%* When the coordinates of the estimated polynomial extremum
      would fall
% outside the coordinates of the points used to constrain the
      estimate.

% Copyright 1993-2004 The MathWorks, Inc.
% $Revision $ $Date: 2004/10/20 17:54:47 $

% get absolute peak pixel
[max_f, imax] = max(abs(f(:)));
[ypeak, xpeak] = ind2sub(size(f),imax(1));

if ~subpixel || ...
    xpeak==1 || xpeak==size(f,2) || ypeak==1 || ypeak==size(f,1) % on edge

return % return absolute peak

else
    % fit a 2nd order polynomial to 9 points
    % using 9 pixels centered on irow,jcol
    u = f(ypeak-1:ypeak+1, xpeak-1:xpeak+1);
    u = u(:);
    x = [-1 -1 -1 0 0 0 1 1 1]';
    y = [-1 0 1 -1 0 1 -1 0 1]';

    % u(x,y) = A(1) + A(2)*x + A(3)*y + A(4)*x*y + A(5)*x^2 +
    %          A(6)*y^2
    X = [ones(9,1), x, y, x.*y, x.^2, y.^2];

    % u = X*A
    A = X\ u;

    % get absolute maximum, where du/dx = du/dy = 0
    x_offset = (-A(3)*A(4)+2*A(6)*A(2)) / (A(4)^2-4*A(5)*A(6))
    ;
    y_offset = -1 / (A(4)^2-4*A(5)*A(6))*(A(4)*A(2)-2*A(5)*A(3));

    if abs(x_offset)>1 || abs(y_offset)>1
        % adjusted peak falls outside set of 9 points fit,
        return % return absolute peak
    end

    % return only one-tenth of a pixel precision
    x_offset = round(1000*x_offset)/1000;

```

```

y_offset = round(1000*y_offset)/1000;
%modded by JP 9/07 to try and get more resolution. Seems
to help.

xpeak = xpeak + x_offset;
ypeak = ypeak + y_offset;

% Calculate extremum of fitted function
max_f = ...
[1 x_offset y_offset x_offset*y_offset x_offset^2 y_offset
^2] * A;
max_f = abs(max_f);

end

```