DEVELOPMENT AND VALIDATION OF A NEW VIDEO EXTENSOMETER SYSTEM FOR COMPRESSIVE MATERIAL TESTING

by

Andrea Felling

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

at

Dalhousie University
Halifax, Nova Scotia
May 2016

Table of Contents

List of Tables

List of Figures

Abstract

Material testing is critical to the development of new alloys, polymers, composites, and other advanced materials. In the last 50 years, the development of video extensometry has made material testing easier than ever before, by allowing researchers to measure sample deformation through image analysis. This thesis details the design, operation, and initial experiments conducted using a new video extensometer system in development under the project name SPECS (Super Portable Extensometer Camera System).

The SPECS system aims to make compressive testing of samples in the presence of barreling more accurate. Rather than reduce friction to assume a non-barreled sample, SPECS simply accounts for barreling by measuring diameter continuously across a sample's height. SPECS does this through a combination of background subtraction and edge detection algorithms. This thesis details the design of the SPECS system, along with the calibration experiments, cold upsetting tests, and finite element modeling used to verify the system.

Operation of the SPECS system is first verified through a series of calibration experiments. With an error on true stress of 4.91%, SPECS error is about 2% greater than stress error seen in cold upsetting tests in the literature. Much of this error is attributed to the quality of the calibration sheet, and improvements to the calibration sheet have the potential for significant reduction to this error.

The verified system is tested against aluminum 2024-T351 in a series of cold upsetting tests. The results of the aluminum alloy tests match well with similar tests in the literature. Data from the aluminum experiments is used to construct and validate a finite element model simulating a cold upsetting test. Diameter data from the finite element simulation matches closely with experimental data.

The long-term goal of SPECS is to create a coupled finite element model/video extensometer tool for fitting advanced material model parameters to experimental data. The experiments in this thesis show that SPECS is capable of collecting accurate stress/strain data which can be used to fit material models for modeling. Future work, therefore, should focus on improving the accuracy of SPECS measurements, and coupling SPECS to the finite element model to make the data fitting procedure iterative and automated.

List of Abbreviations and Symbols Used

A_i	instantaneous cross-sectional area of deformed sample
C_{dots}	factor by which nominal calibration sheet dot spacing is incorrectly estimated
D	sample diameter
H	sample height
L_i	instantaneous deformed length of sample
L_o	original undeformed length of sample
M	magnification of an object viewed through camera
P	applied load
R_{FSR}	resistance of force sensing resistor
R_{fixed}	resistance of fixed resistor placed in series with force sensing resistor
WD	working distance between camera and target
X_i	X coordinate of a detected edge
$X_{i,j}$	dependent variable j at data point i, used in linear regression
Y_i	Y coordinate of a detected edge
ΔD	% change in inner diameter for friction ring tests
ΔH	% change in height for friction ring tests
Δx	horizontal spacing between calibration sheet dots
Δy	vertical spacing between calibration sheet dots
Δ_{dots}	spacing between dots on calibration sheet
β	matrix containing variables to be fit by linear regression
δ_{HG}	platen displacement measured by height gauge
δ_S	platen displacement measured by SPECS
ϵ_{eng}	engineering strain
ϵ_{true}	true strain
μ	coefficient of friction
σ_{eng}	engineering stress
σ_{true}	true stress

b	offset used for linear fitting
e	error
f	camera focal length
f_{dots}	correction factor applied to correct calibration sheet dot spacing
k	power law strength coefficient
m	slope used for linear fitting
n	power law hardening coefficient
y_i	independent variable at data point i, used in linear regression

ASTM	American Society for Testing and Materials
DAQ	Data Acquisition. Often used to refer to an entire Data Acquisition module.
DIC	Digital Image Correlation
FEM	Finite Element Modeling. Alternately, Finite Element Method
FSR	Force Sensing Resistor
PM	Powder Metallurgy
SPECS	Super Portable Extensometer Camera System
VI	Virtual Instrument, a type of LabVIEW script

Acknowledgements

I would like to express my appreciation for the wonderful professors, technicians and support staff of the mechanical engineering department whose support and expertise were invaluable in completing this work.

My sincerest gratitude to my thesis supervisor, Dr. Darrel Doman, for his amazing mentorship and endless support throughout my degree. Thanks as well to my thesis committee, Drs. Clifton Johnston and Stephen Corbin for their valuable insight into my work.

Thank you to Angus MacPherson, Mark McDonald, Albert Murphy and for their time and expertise in manufacturing the equipment and test samples for this project

Finally, thank you to my friends and family, whose love and encouragement have kept me motivated throughout this degree.

The work in this thesis was supported by a grant from the National Sciences and Engineering Research Council of Canada.

Chapter 1: Introduction

1.1 Background

A thorough understanding of the mechanical properties of materials is critical in mechanical engineering design. Most engineering design concerns itself with materials subjected to elastic, or non-permanent deformation. Under such conditions, an approximated or engineering stress (which does not account for a change in cross section) is used. The deviation between this engineering stress and true stress, which accounts for a changing cross section, is insignificant for small strains. When a material undergoes large straining (such as massive plastic deformation), the deviation between true and engineering stress becomes significant, and the change in dimension must be accounted for.

Standard compressive test methods such as those detailed in [1] and [2] measure a sample's vertical deformation in response to an applied load, and use an assumption of volume consistency (volume remains constant throughout testing) to calculate the test sample's horizontal dimensions. In the case of standard metallic alloys, this is a reasonable assumption to make, as the material is already in its most dense possible condition. This assumption breaks down, however, when working with materials produced through powder metallurgy (PM) manufacturing techniques. In the manufacturing process of a PM part, residual porosity is inevitably incorporated into the material. For some applications, such as oil-infused bearings, this porosity is desirable [3]. However, these pores also allow for a change in volume under compression, invalidating the key assumption in standard material testing.

The first obvious solution - measure the sample's width during testing - has issues of its own. Firstly, there is the matter of friction. As shown in Figure 1.1, when friction acts between a cylindrical test sample and the tooling compressing it, material is unable to spread freely at the tool-sample interface, leading resulting in the sample barreling, or bulging, around its middle. Measuring the width of the sample at multiple points through traditional means such as clip-on extensometers is impractical. The physical size of the extensometers mean that only a very small number of measurement points could be used. Instead, test methods prescribed in ASTM standards [1] advise researchers to avoid the barreling effect altogether

by eliminating as much friction as possible. The extra preparation to adequately lubricate the samples and test equipment add extra time and extra room for error to the material testing process.

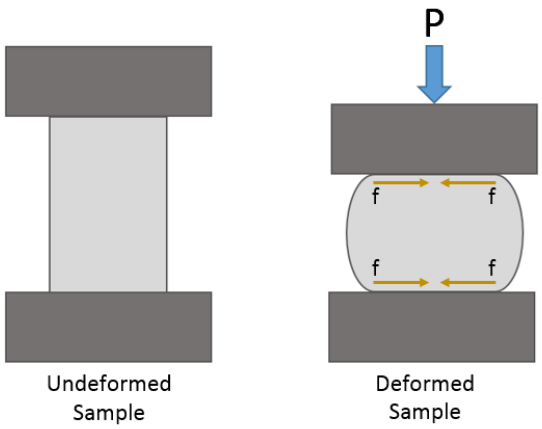


Figure 1.1: Barreling as a result of friction between tooling and test sample

Even if barreling is mitigated, traditional extensometers have another critical drawback when testing materials through plastic deformation. Most extensometers are relatively delicate in construction. Many manufacturers recommend removing the extensometer from a sample before reaching breaking stresses, to protect the instrument from being damaged in the sudden release of energy. This results in a limitation on the amount of compressive strain that can be safely imparted in a sample.

Video extensometery has the potential to mitigate these issues. Simply put, a video extensometer is a device which uses cameras and image analysis to measure the dimensions of a test sample. to Unlike a clip-on extensometer, which can only measure strain in a single discrete location, a video extensometer can measure diameter across the entire height of a sample. This allows for the sample's entire curvature to be measured, eliminating the need to reduce the effects of friction and barreling. Additionally, a video extensometer system is also able to measure data right up to the breaking point of a material. As there is no contact between the extensometer and the sample, there is no risk of damaging delicate equipment during material failure.

1.2 Objectives

1.2.1 Thesis Objectives

The focus of this work is the development, validation, and application of a new video extensometer system developed under the project name SPECS (Super Portable Extensometer Camera System). By using image analysis to measure the deformation of a test sample, this system attempts to address the issues that arise from measuring diameter through calculations or contact methods. Three sets of experiments were conducted with respect to this primary objective:

1. Calibration experiments, focusing solely on the accuracy of measures of sample size and displacement. These experiments were conducted by comparing SPECS measurements to precisely measured vertical displacements and high accuracy calibration blocks.

2. Validation experiments, conducted on Aluminum 2024-T351 alloy. In these experiments, aluminum samples were compressed to failure, and SPECS measurements were used to develop true stress, true strain diagrams. Experimental data was compared to well established data sets on the same alloy from the literature.

3. Finite Element Modeling (FEM) experiments. In these experiments, the results of the validation experiments were used to develop and validate a finite element simulation of a standard compression test.

1.2.2 Long-Term Objectives

The work in this thesis builds the foundation for two long-term goals of the SPECS system:

1. Development of a coupled SPECS/FEM tool for fitting finite element material model parameters.

2. Testing of materials which violate volume consistency assumptions used in other material test methods.

The first long-term goal - the development of a coupled SPECS/FEM tool - is intended to reduce the effort and time required for fitting advanced material models to new uncharacterized materials. The tool will couple the SPECS video extensometry system developed in this thesis with finite element modeling and an optimization algorithm. Starting with parameters estimated from a sample's force-displacement curve, the complete tool will simulate a compression test using finite element modeling. Results from the finite element model for sample height, curvature, diameter and other metrics will be compared to experimental results from SPECS. The optimization algorithm will then use this information to iterate new material parameters. This process will repeat itself until a set of parameters which minimizes error between SPECS and the finite element model is found.

The second long-term goal for SPECS is the study of materials which violate the assumption of volume consistency used in standard test methods. For example: porous materials such as powder metallurgy parts or metal foams exhibit varying degrees of volume inconsistency, as they have air incorporated in their structures which can be evacuated or compressed during material deformation. To fully characterize these materials, diameter data cannot be estimated, it must be directly measured. A video extensometer such as SPECS is uniquely suited to this application, as diameter data is captured directly rather than estimated from other parameters. This goal ties into the previous goal, as a coupled FEM-SPECS tool will be invaluable for fitting experimental data to a material model which can capture the sample's changing volume throughout testing.

Chapter 2: Literature Review

2.1 Video Extensometers in Material Testing

The term "video extensometer" has been in use for several decades to describe a broad spectrum of measurement devices that use image analysis to conduct strain measurements. The earliest precursors to modern video extensometers are first seen in the late 1960's and early 1970's in bio-medical research fields [4, 5]. These early video extensometers were developed as a response to unsatisfactory and impractical mechanical methods previously used in the field. The relatively high compliance of living tissues, for example, meant that contacting methods such as calipers or strain gauges interfered too much with normal motion and deformation of the tissue [4]. For researchers studying living tissue in-vivo, purely mechanical measures would also necessitate unnecessary surgery. Making these measurements indirectly via image analysis was possible with the use of various x-ray techniques and fluorescent dyes, but doing frame-by-frame analysis manually was tedious and inaccurate.

To make the measurement process more precise and automated, researchers began developing tools to convert television signals into computerized measurements. Television systems at the time produced images by scanning a cathode ray across the screen in a series of horizontal lines. Using digital logic gates, researchers measured the light intensity while the television scanned within a region of interest, and allowed an analog output voltage to build up whenever the light intensity was below a given threshold. The magnitude of this output voltage was directly proportional to the width of the dark silhouette being measured in the television image [4].

As digital photography developed and advanced, video extensometry did as well. Starting in the 1980s, measurement systems which analyzed fully digital images began to be developed [6]. By the early 1990s, commercially available video extensometers began to be developed by material testing companies such as Instron [7].

2.1.1 Marker-tracking Video Extensometers

The simplest modern video extensometers function very similarly to both traditional contacting extensometers early television based measurement systems. Broadly classifiable as marker-tracking extensometers, these extensometers rely on detecting a small number of high-contrast marks made on a test sample, typically lines or dots. In the most basic implementation, a pair of markers can be used almost identically to a contact extensometer: the initial distance between the markers defines a gauge length, and the change in this gauge length over the course of a test is used to calculate strain [7, 8]. More complex implementations use a larger number of markers, allowing the systems to capture a more complete picture of changing strains across the sample. The seven dot configuration seen in Figure 2.1 is particularly useful in polymer research [9, 10, 11]. This configuration has the advantage of tracking both vertical and horizontal deformation. The ability to track horizontal deformation is critical, as some polymers exhibit volume change during deformation, so diameter cannot simply be calculated from a change in height [10].

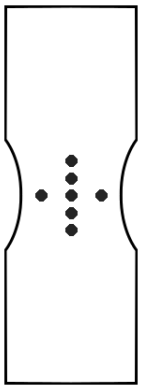


Figure 2.1: Seven dot configuration used in marker-tracking extensometers

The simplicity of marker-tracking video extensometer systems can be seen as an advantage. The image processing required to locate the centers of a small number of dots is computationally inexpensive. This allows image data be processed in real time, and the resulting strains can be used to control testing systems. For samples with complex stress-strain curves, this means that objectives such as constant strain rate loading are possible [9].

2.1.2 2D Digital Image Correlation Extensometers

2D Digital Image Correlation (2D-DIC) video extensometers are among the most frequently discussed systems in the literature, and are sold commercially as an advanced alternative to simpler mark-tracking systems. In a typical DIC implementation, a thin, planar sample is first prepared with a random, matte pattern of speckles. An undeformed image of the sample is first taken as a reference image, and then a load is applied to the sample. This reference image is compared to images taken of the sample deformed under load. The DIC system then analyzes small sub-regions of the original reference image, known as subsets. Each subset from the original image is deformed and translated, and the results are compared to the corresponding pixels in the deformed image. The DIC system finds an optimal translation and deformation for each subset which produces the best possible match with the actual deformed image. By repeating this process on small subsets across the entire image, strains can be computed across the entire sample. [12, 6]. The process is shown schematically in Figure 2.2 below.

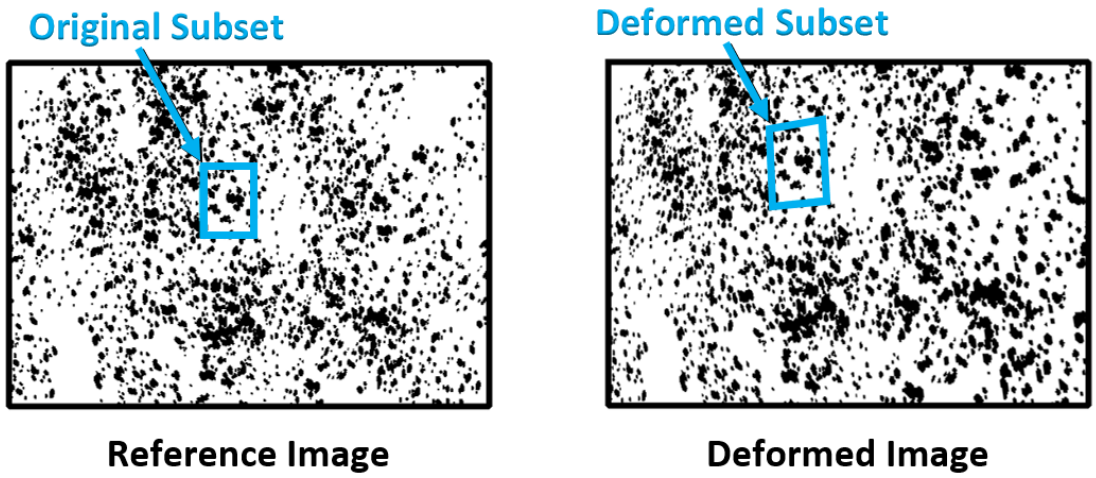


Figure 2.2: Transformation of a subset in 2D-DIC

The application of the initial speckle pattern is critical for accurate results in 2D-DIC. A recent paper which sought to investigate sources of error in 2D-DIC found that the type of speckle pattern used had a noticeable impact on error [13]. In particular, a poor quality speckle pattern can lead to some measurement points having *no* correlation results. The authors further note that the quality of the speckle

pattern is dependent on the operator, and that sometimes many attempts are needed to produce a pattern appropriate for DIC.

When 2D-DIC was first being developed in the early 1980s, the computational time needed to perform correlation was extensive, taking upwards of half an hour to perform a single match. With the exponential increases in computing power since then, the processing speed of 2D-DIC is now near-real-time speed [12]. The speed and relative ease-of-use associated with 2D-DIC make it an attractive option for many researchers.

However, 2D-DIC has one very critical limitation. As the name implies, 2D-DIC is only applicable to thin, planar samples experiencing almost pure plane stress [6]. Out-of-plane motion leads to falsely strain measurement error, as an object's apparent size in an image is proportional to its distance from the camera, as shown in Figure 2.3. Thus, motion towards the camera would make the speckles (and the spacing between them), bigger, leading to an artificially high strain measurement. Likewise, motion away from the camera would reduce the apparent distance between speckles, leading to an artificially low strain measurement.

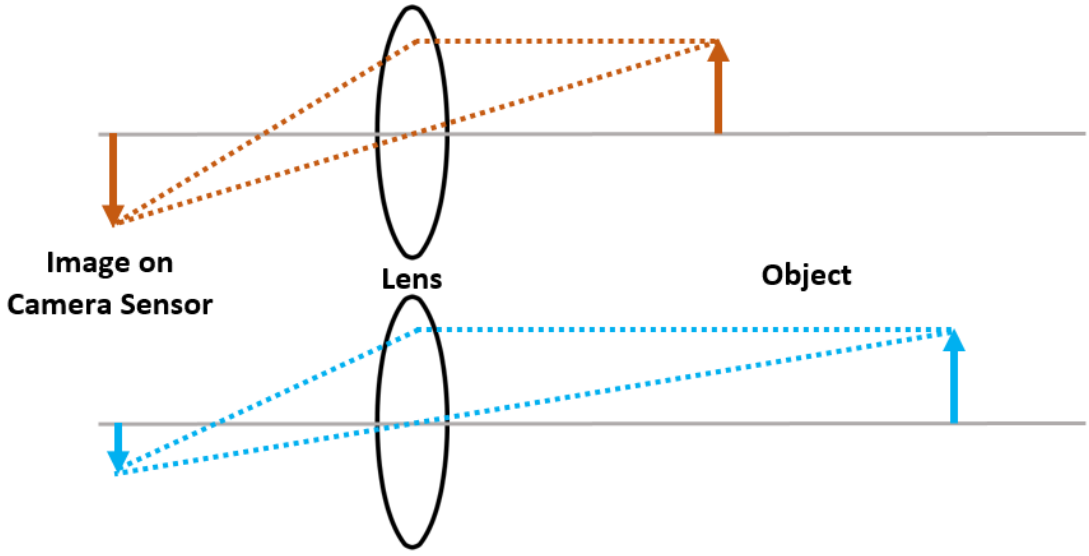


Figure 2.3: Ray diagram showing the effect of object distance on apparent image size

2.1.3 3D Digital Image Correlation Extensometers

Multi-camera 3D Digital Image Correlation (3D-DIC) systems operate under the same principles as 2D-DIC, but with two separate cameras set up in a stereoscopic configuration. With the system collecting images from two slightly different perspectives, displacements are measured in full three-dimensional coordinates [6]. This allows 3D-DIC to be used on samples which have significant out-of plane deformations, which is not possible with a 2D-DIC system.

While 3D-DIC overcomes many of the shortcomings associated with a 2D-DIC system, it is not without its own limitations. While 3D-DIC systems are capable of mesuring out of plane deformation, particularly complex curvature or particularly severe deformation may lead to blind spots, where part of the sample is occluded in the view of one or both cameras [14]. Furthermore, a 3D-DIC system operates only where the fields of view of its two cameras overlap. Adjustments made to camera spacing in order to accommodate large objects could lead to a reduction in the effective area covered by the cameras.

Recent developments within 3D-DIC attempt to address these limitations by considering a multi-camera (i.e. more than two) DIC system [14]. In such a system, each possible pair of cameras is considered as an independent stereo-vision system. This can dramatically reduce the presence of blind spots, as details are only completely occluded if they are occluded in the view of every possible camera pairing. A greater confidence in measurements can also be obtained for details seen by more than one camera pair. However, there is a practical limit to how many cameras could be implemented in this sort of system. The number of possible camera pairs, and therefore the number of independent DIC processes required, increases steeply with each additional camera as seen in Figure 2.4.

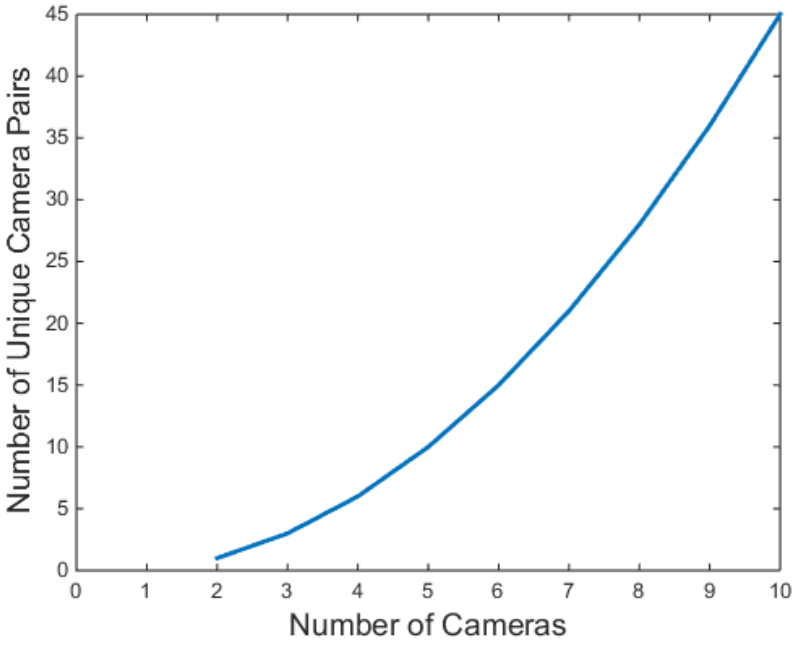


Figure 2.4: Unique camera pairs for a given set of cameras

2.2 Compression Testing and the Cold Upsetting Test

Uniaxial compression testing is of great interest in the field of constitutive material modelling and plastic deformation prediction. In particular, the cold upsetting test is one of the fundamental tests used in many papers on the subject, especially papers which are concerned with modelling ductile fracture. This provides a large body of work with comprehensive data sets to compare the results of these initial experiments to.

The largest body of work in the area of fracture prediction appears to come from the Impact and Crashworthiness Laboratory at MIT. Researchers in this lab have produced a number of papers predicting and modeling fracture behavior in Aluminum 2024-T351, a heat-treated aluminum alloy used in aerospace and biomedical applications [15]. In particular, a paper by Wierzbicki, Bao, Lee and Bai compiles data from 15 different experiments on Al2024-T351, including 4 cold upsetting experiments on cylinders with aspect ratios (d_o/h_o) ranging from 0.5 - 1.5 [16]. By taking a range of aspect ratios, Wierzbicki et al. were able to extrapolate their experimental results to an infinite-height cylinder, which theoretically would experience no barreling.

This idea that an infinitely tall cylinder would experience no barreling was initially

proposed by Johnson and Mellor [17], who state that when cylinders of different heights but equal diameter are compressed, the degree of barreling is dependent on the original length, and is least for the longest cylinder. They posit that for an infinitely tall cylinder, the end effects that lead to barreling would be negligible. Their conclusions are supported throughout the literature, with numerous authors noting that increasing the height of a cylinder for a given diameter significantly increases the radius of barreling [18, 19, 20, 21]. However, using extremely long, narrow samples to eliminate barreling is impractical and dangerous, due to concerns of buckling and instability in such a large column.

In an investigation that aimed to establish universal fracture criteria for ductile metals, Khan and Liu revisited the body of work from Weirzbicki et al. through a combination of torsion, tension and compression experiments on Aluminum 2024-T351 round bar [22]. In addition to their main experiments, Khan and Liu also developed stress strain curves for samples compressed in both the diametral and axial directions to prove their material behaved isotropically. Khan and Liu's experiments provide a second, independent data set to compare to Wierzbicki et al. and the experiments detailed in this report.

Further experiments on Al2024-T351 were conducted by Seidt and Gilat[23], with a focus on studying sensitivity of material response to parameters such as strain rate and temperature. In their tests, the authors tested samples cut from a sheet of rolled Al2024-T351 in tension, compression, and torsion. The key conclusion which comes from this work is that there is essentially no sensitivity to strain rate for rates between $10^{-4}s^{-1}$ and $5 \times 10^{3}s^{-1}$. For a typical hydraulic test frame, which requires anywhere from a few dozen seconds to several minutes to compress a test sample, all practically achievable strain rates fall within this range.

2.3 The Barreling Effect and Friction Ring Testing

The barreling effect has been extensively studied, for both how to predict the extent of barreling and how to reduce its effect. An ASTM Selected Technical Paper discussing procedures for compressive testing on metals outlines several common approaches used throughout the literature [2]. The first and most obvious approach is to lubricate the sample and tooling to reduce the coefficient of friction at the interface. The ASTM paper found that for Teflon film lubricants, coefficients of friction as low

as $\mu = 0.04$ could be achieved and the barreling effect all but eliminated. It was also found that concentric grooves on the top and bottom face of a cylindrical test sample could be used to help retain the lubricant, improving the effect of the lubricant even under severe deformations. The report ultimately concludes, however, that it may simply be easier and less time consuming to simply account for friction using a friction ring test.

The friction ring test uses a clever application of the barreling effect to measure friction between a material and tooling during a forming operation. First established by Kunogi in 1956 [24] and made practical by Male and Cockcroft in 1964 [25], the friction ring test is still utilized by researchers today through both experimental work and Finite Element Modeling (FEM) [26, 27].

In a friction ring test, a squat ring of the material of interest is compressed so it plastically deforms. Under perfect, frictionless conditions, both the ring's inner and outer diameter expand due to Poisson effects. With the introduction of friction, however, the inner diameter of the ring barrels out, reducing the amount by which the inner diameter grows, as seen in Figure 2.5. At a critical value of friction, $\mu = 0.055$, the inner diameter of the ring actually shrinks with compaction, as seen in Figure 2.6. The sample's dimensions are measured before and after testing to calculate the percentage change in height (ΔH) and percentage change in inner diameter (ΔD).

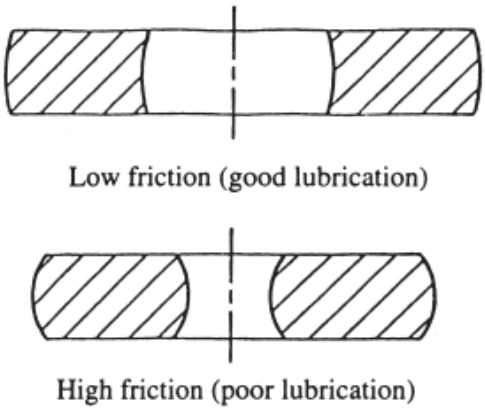


Figure 2.5: Deformation of a friction ring under varying friction conditions (from [26])

The resulting coefficient of friction can be obtained by comparison to a set of curves like those seen in Figure 2.6, or can be obtained through the set of empirical

equations below developed by Male and Cockroft through their work in developing a practical application of the friction ring test [25]. The authors note that these equations are valid for values of $0.055 \leq \mu \leq 0.57$ and $20\% \leq \Delta D \leq 60\%$, and only for rings geometrically similar to the ones used in their experiments (0.75" OD, 0.375" ID, 0.250" height).

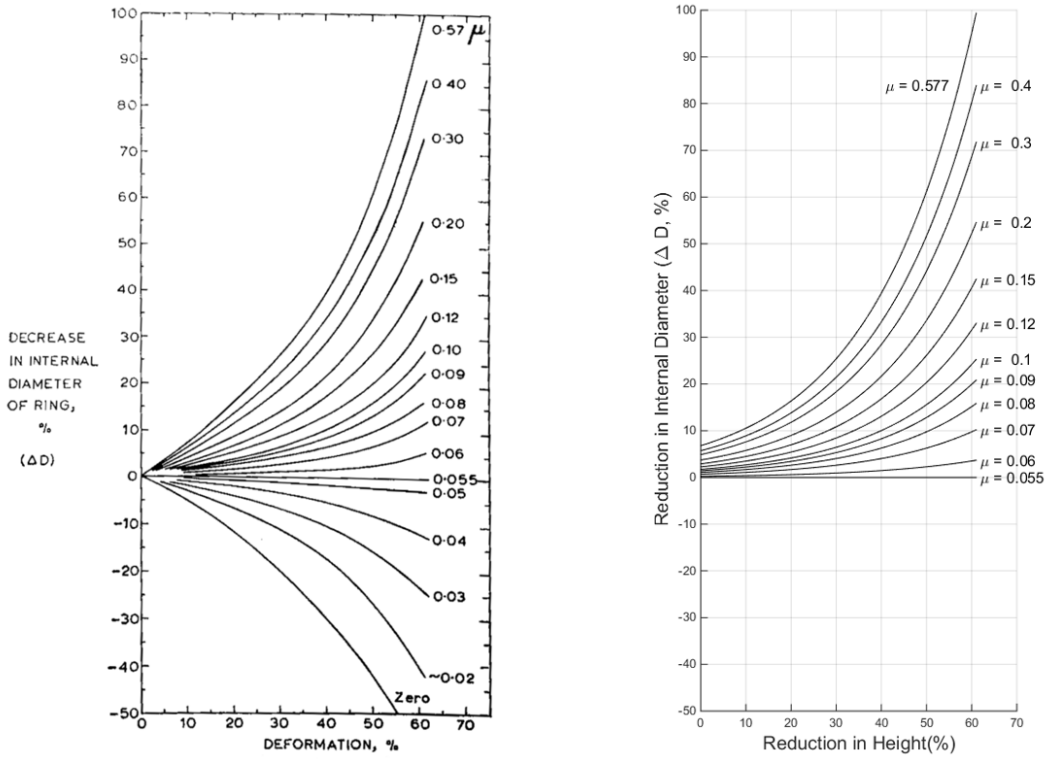
The percentage decrease in inner diameter (ΔD) is given by:

$$\Delta D = m \ln\left(\frac{\mu}{0.055}\right)$$

where m is given by:

$$\ln m = (0.044 \times \% \text{ height deformation}) + 10.6$$

An attempt was made to verify these equations by using them to reproduce Male and Cockcroft's calibration curves for the specified ranges of ΔD and μ. As written, the equations do not produce similar curves at all, suggesting that a 60% reduction in height should produce a physically impossible 300,000% reduction in diameter. However, replacing 10.6 with 1.06 in the equation to calculate m produces curves which match very closely with the source curves, as seen in Figure 2.6, suggesting that this may, in fact, be a typo in the original work.



(a) Male and Cockcroft's Original Friction Ring Calibration Curves [25]

(b) Curves reproduced from Male and Cockcroft's
Equations (correcting 10.6 to 1.06)

Figure 2.6: Male and Cockroft's friction ring calibration curves

Chapter 3: System Overview

3.1 Test Frame and Platens

All prototyping, and validation of the SPECS system was conducted using an Instron RD 600 3 MN hydraulic compression test frame, as seen in Figure 3.1. Commonly referred to simply by brand name (i.e. "an Instron"), a hydraulic test frame controls a large piston inside its test chamber through hydraulic pressure. The piston applies load to test samples through various specialized fixtures, or through flat plates known as platens.

As seen in Figure 3.1, additional equipment was installed inside the test frame to create a controlled and consistent optical environment. Inside the test chamber, two banks of LED lighting strips are mounted to the side walls to provide bright illumination of the entire test chamber. The rear door is covered on the inside by a hanging sheet of colored card stock, which creates a flat, undetailed background to view tests against. The sheet is held in place with magnetic clips, allowing the background sheet to be easily replaced. Finally, a plastic camera bracket was installed inside the front door to provide a stable mounting point for the camera.

By design, the Instron test frame was left to operate independently of the SPECS system. One of the key design criteria for SPECS was portability, so the less SPECS was integrated with one specific test frame, the easier it would be to port the system to other testing setups. During testing, the Instron test frame is controlled using commercial material testing software on its own dedicated computer. In addition to controlling the tests, the software records the piston's load and position in a timestamped database, which can be exported for postprocessing.

During testing, samples are loaded using steel spacers platens placed above and below the test sample. To aid in optical tracking, specially designed platens were produced. One set of platens was made from 4140 steel, turned on a lathe to be flat and parallel. A pair of holes were reamed directly opposite eachother on the sides of the platens, to serve as a mounting location for two 1/2" stainless steel tooling balls. When viewed through a camera, the 1/2" tooling balls appear as nearly perfect circles, providing an easy target for optical tracking.

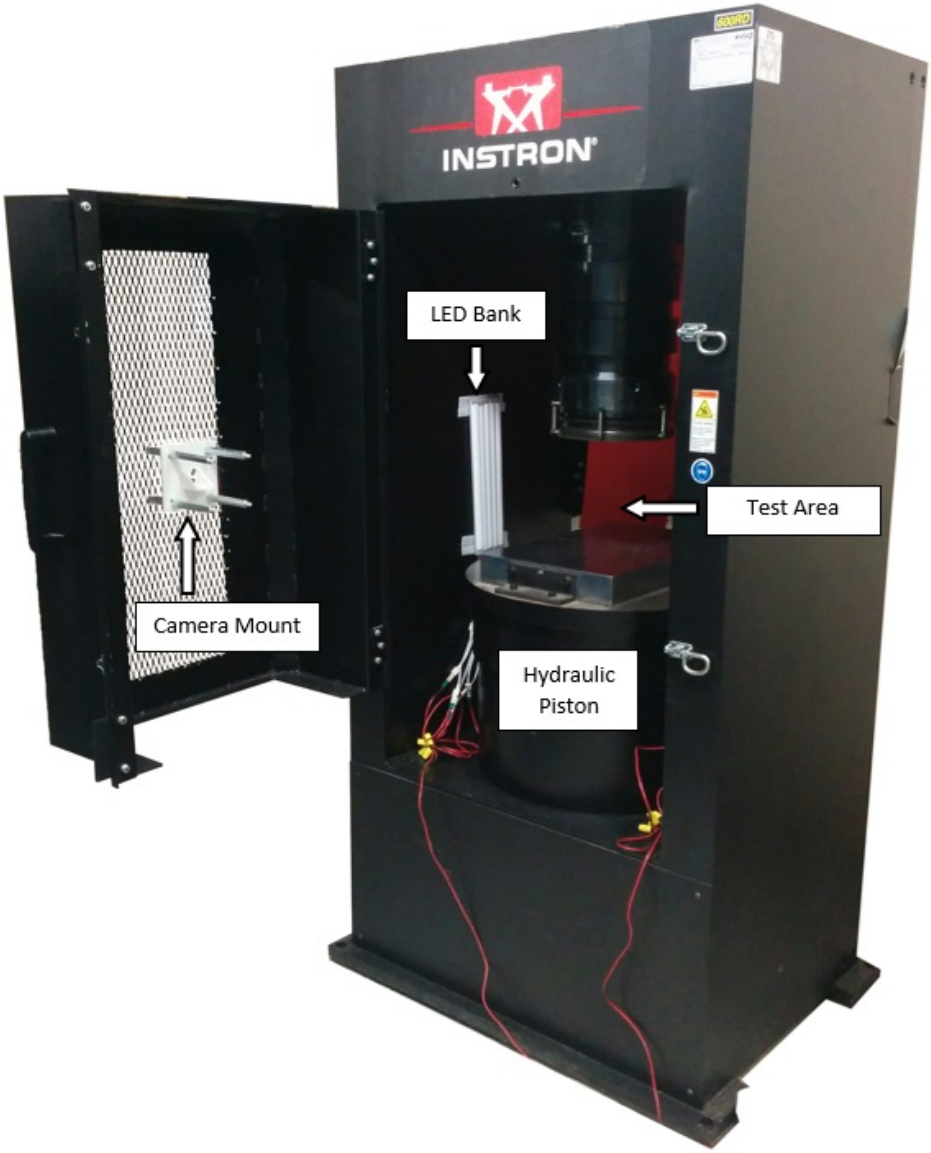
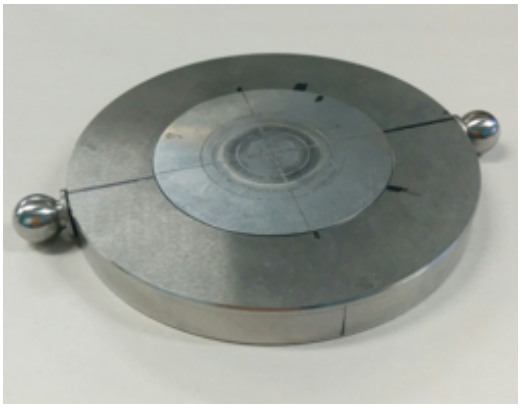


Figure 3.1: Instron RD 600 Hydraulic Test Frame

centering ring and a 4140 steel platen are shown (with tooling balls) in Figure 3.2.



(a) Hardened stainless steel platen in centering ring

(b) 4140 steel platen

Figure 3.2: Steel compression platens

3.2 Camera and Electronics

SPECS uses a Point Grey Research Flea3 camera outfitted with a 3.5 mm low distortion lens lens for image acquisition. The camera captures full color images at a resolution of 4096 x 2160 pixels, at a maximum rate of 21 frames per second. The camera is connected directly to a data acquisition laptop via a USB3 cable, which serves a dual purpose of powering the camera and providing a high speed data connection to save acquired images for future analysis. The camera is also connected to a cDAQ-9174 Data Acquisition (DAQ) chassis through two digital lines known as the *trigger* and *strobe* lines.

The trigger line is an input line to the camera, which operates on a 3.3 V digital signal. When in a triggered operation mode, the camera does not expose images automatically. Instead, it waits in a standby mode for the trigger line to drop from HIGH (3.3 V) to LOW (0 V). Upon detecting the falling edge, the camera begins to immediately expose an image. When the exposure finishes, the camera returns to standby mode, waiting for the next falling trigger [28].

The strobe line is an output line from the camera, which also operates on a 3.3 V digital signal. While the camera is on standby, the line is kept HIGH (3.3 V). When the camera begins to expose an image, the strobe line falls to LOW (0 V), and remains LOW for the duration of the exposure.

A force sensing resistor (FSR) connected to the DAQ chassis is used to synchronize

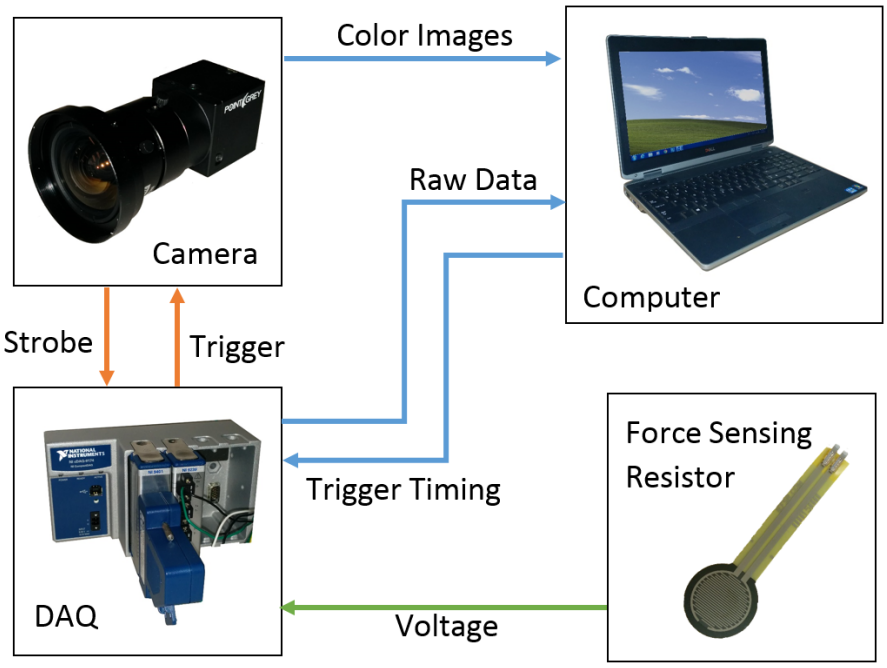


Figure 3.3: System hardware overview and data flow

timestamps between data collected by SPECS and data collected by the test frame. Similar to a strain gauge in construction, an FSR is a resistor built on thin film, which is embedded in a flexible membrane. As the membrane is deformed, the resistance changes dramatically. By including the FSR as one leg of a half bridge, this change in resistance is easily measured by the DAQ as a voltage. The FSR is glued to the start button of the hydraulic test frame so that a voltage pulse is generated whenever the button is pressed. This provides a common reference point which exists in both the SPECS data and the test frame data.

3.3 Data Acquisition Software

A laptop computer running LabVIEW software was used to drive the SPECS system. A VI (LabVIEW script) runs continuously on the laptop during testing to control the camera and record data. Whenever the camera exposes a new image, the VI saves the image, strobe line voltage, and FSR voltage to an output data folder. Figure 3.4 shows a flowchart describing the data acquisition software.

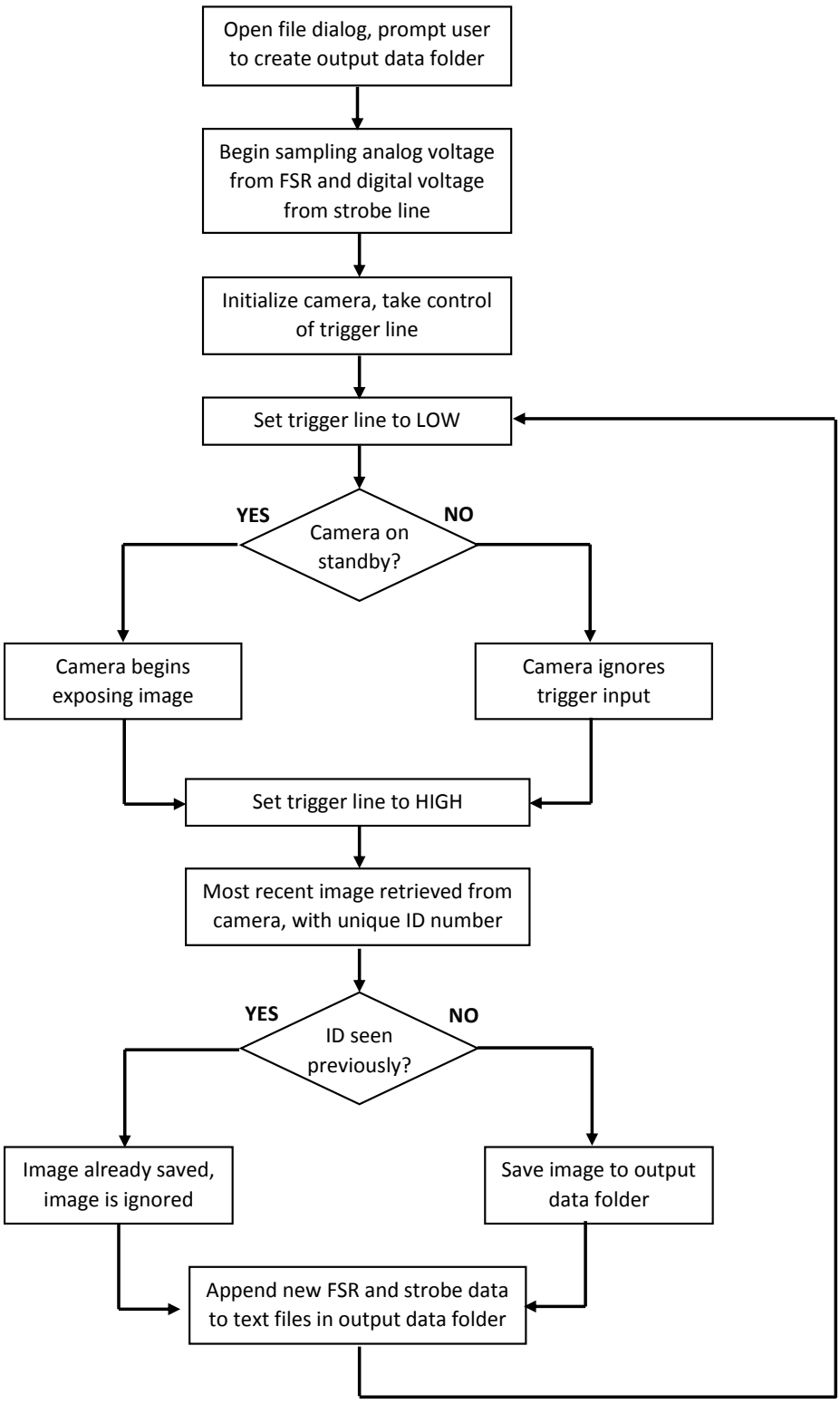


Figure 3.4: Data Acquisition Software VI

3.4 Postprocessing Software

The SPECS system uses a combination of background subtraction, edge detection and circle detection algorithms to isolate the silhouette of a test sample from a surrounding image. When applied to cylindrical samples, this silhouette provides the sample's profile at mid-plane. All image processing for SPECS is done using a postprocessing VI which batch processes images sharing a common background and calibration template.

The LabVIEW image processing outputs a series of text files containing the (x,y) coordinates of all detected edges and circles in both pixels and real-world coordinates. MATLAB software then combines these edge and circle detections with loading data from the instron, and timing data from the DAQ software. Using these inputs, the MATLAB code calculates stress, strain, and diameter for the test sample. Figures 3.5 and 3.6 give a flowchart overview of the LabVIEW and MATLAB postprocessing softwares.

3.4.1 LabVIEW: Image Calibration

Before any image processing is done, test images are first calibrated, so that positions and distances can be reported in real-world units. The calibration process is relatively simple, and only needs to be done once each time the camera is moved. SPECS uses a microplane calibration technique, which provides a one-to-one mapping of pixel coordinates to real world coordinates [29, 30].

To calibrate the camera, a flat calibration target printed with hundreds of calibration dots, which is placed in the center of the test chamber. An example of the calibration grid used for SPECS can be seen in Figure 3.7 The spacing between the dots is measured in real-world coordinates. This measurement is passed to a calibration learning VI, along with an image of the calibration sheet.

To learn a calibration map from the image, a region of interest is drawn around the calibration target, and thresholding is applied to isolate the calibration dots. Each calibration dot is taken as a reference point. As seen in Figure 3.8, each reference point has both a pixel location (located at the center of the dot), and a real-world location (calculated from the inter dot spacing Δx and Δy. Pixel to real-world mappings are then computed for all pixels in the image, using the surrounding

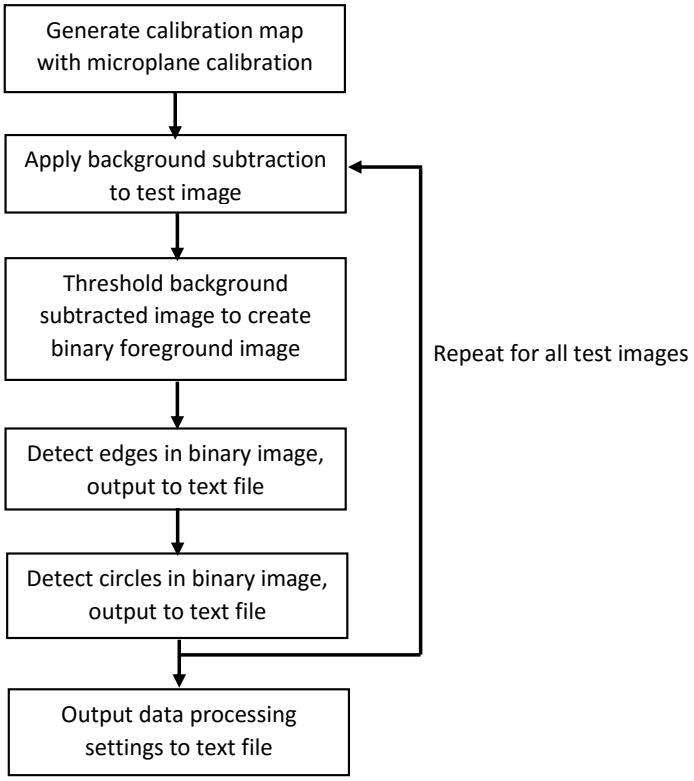


Figure 3.5: LabVIEW postprocessing software

reference points to estimate the real-world location of each pixel.

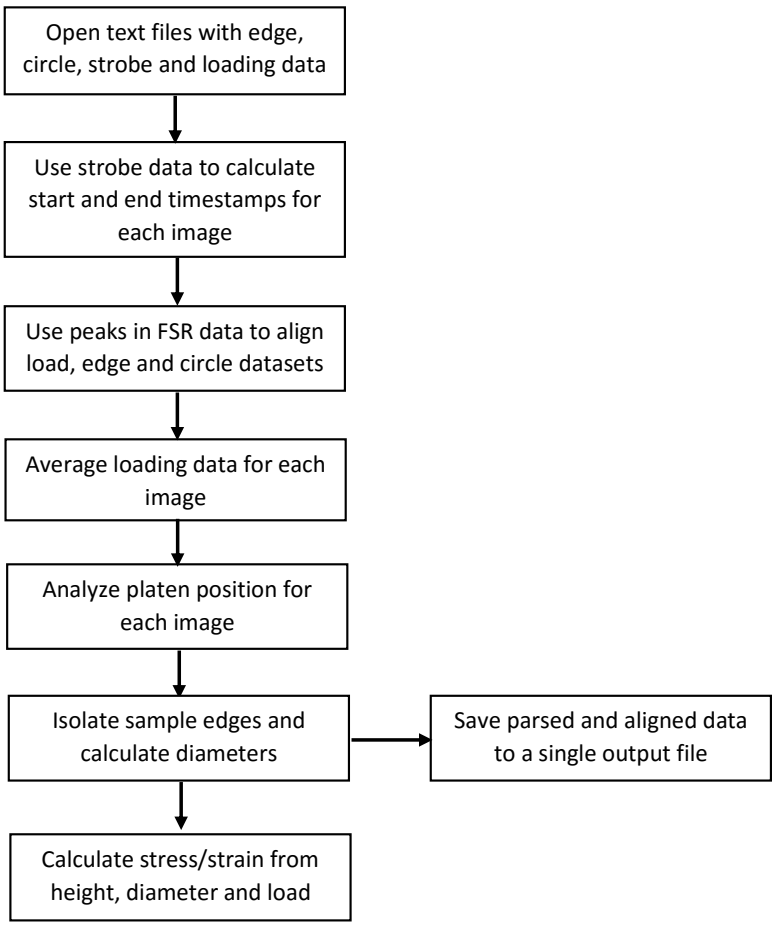


Figure 3.6: MATLAB postprocessing software

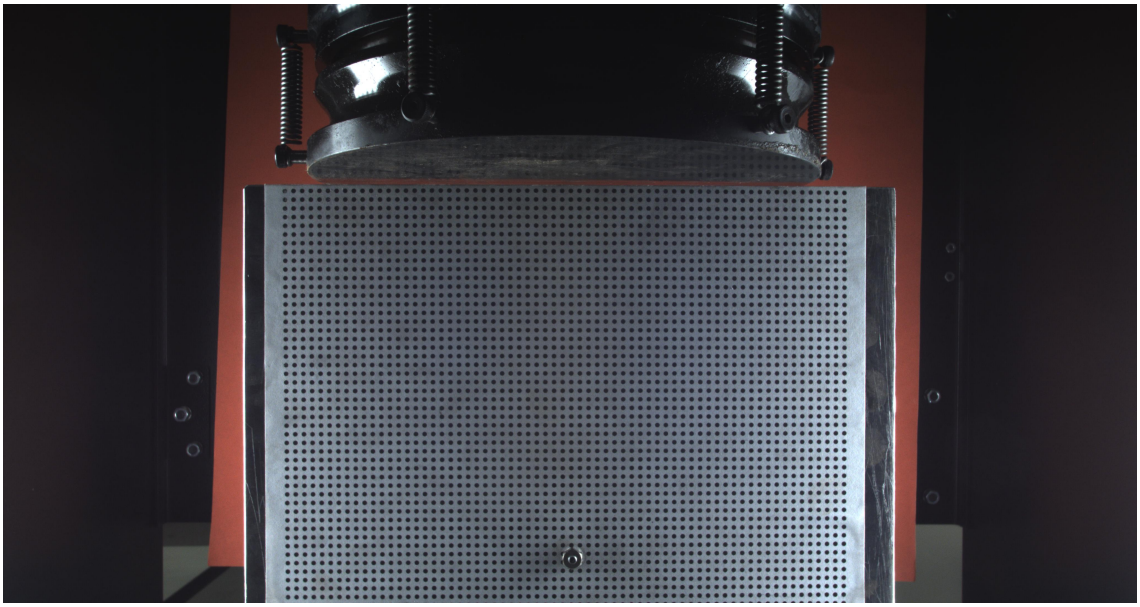


Figure 3.7: A typical calibration remplate image

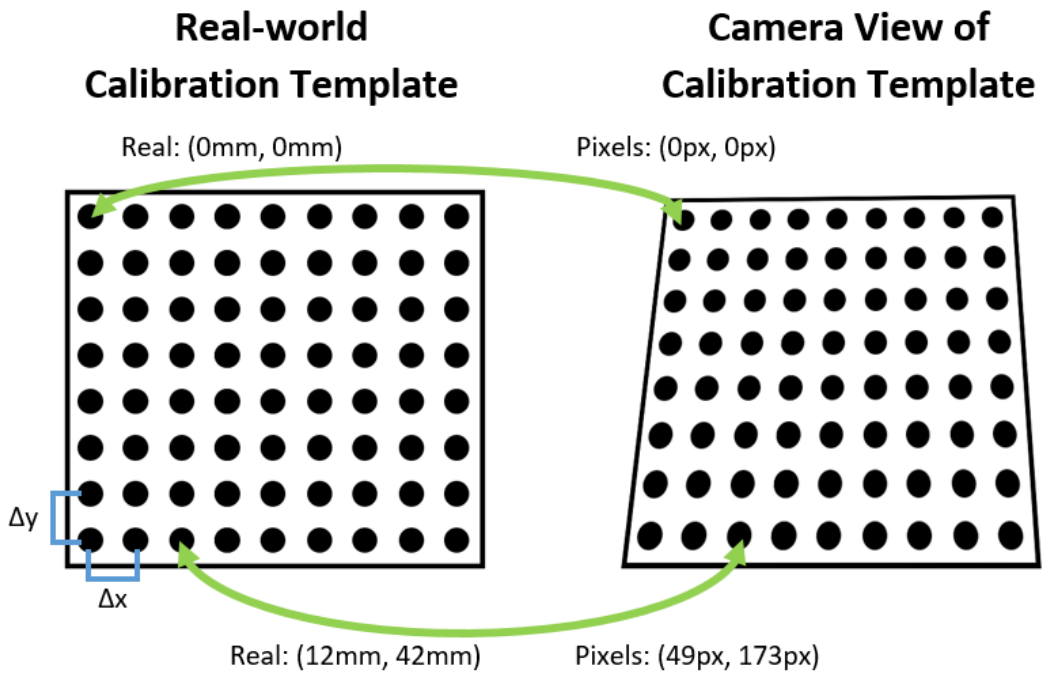


Figure 3.8: Matching real-world and pixel coordinates for calibration reference points

3.4.2 LabVIEW: Background Subtraction

The background subtraction algorithm is a computer vision technique used to extract foreground elements from the surrounding background in an image. It does so by comparing each image of interest to an empty image of the same scene. An overview of the background subtraction process can be seen in Figure 3.9.

The SPECS implementation of background subtraction starts by splitting each image into its three component color planes: red, green and blue. Each of these component images is stored as an array of unsigned, 8-bit integers. A background image showing just the empty test chamber is likewise split into its component planes. For each color plane, the background image is subtracted from the image of interest, and the absolute value is taken. The absolute difference arrays for the three color planes are then summed into one single image, which serves as a map of how different each pixel in the test image is from the corresponding background image pixel.

A threshold value, selected by the user, is then used to convert the summed absolute difference image from an 8-bit grayscale image into a binary image. Pixels values below the threshold are considered as pixels from the background, and are rounded of 0. Pixels with values above the threshold are considered to be pixels from the foreground, and are rounded to 1.

The bright red background seen in Figure 3.9 was chosen deliberately to provide high contrast for the Aluminum 2024-T351 test samples. In general, it was found that brightly colored backgrounds performed better in background subtraction than various grayscale shades. White backgrounds in particular performed poorly, as they caused reflections on the sample to register as background pixels, which were subsequently subtracted. In future experiments, a preliminary test of background colors is recommended for best results, as samples with red hues like copper and brass may not work well with the background currently in use.

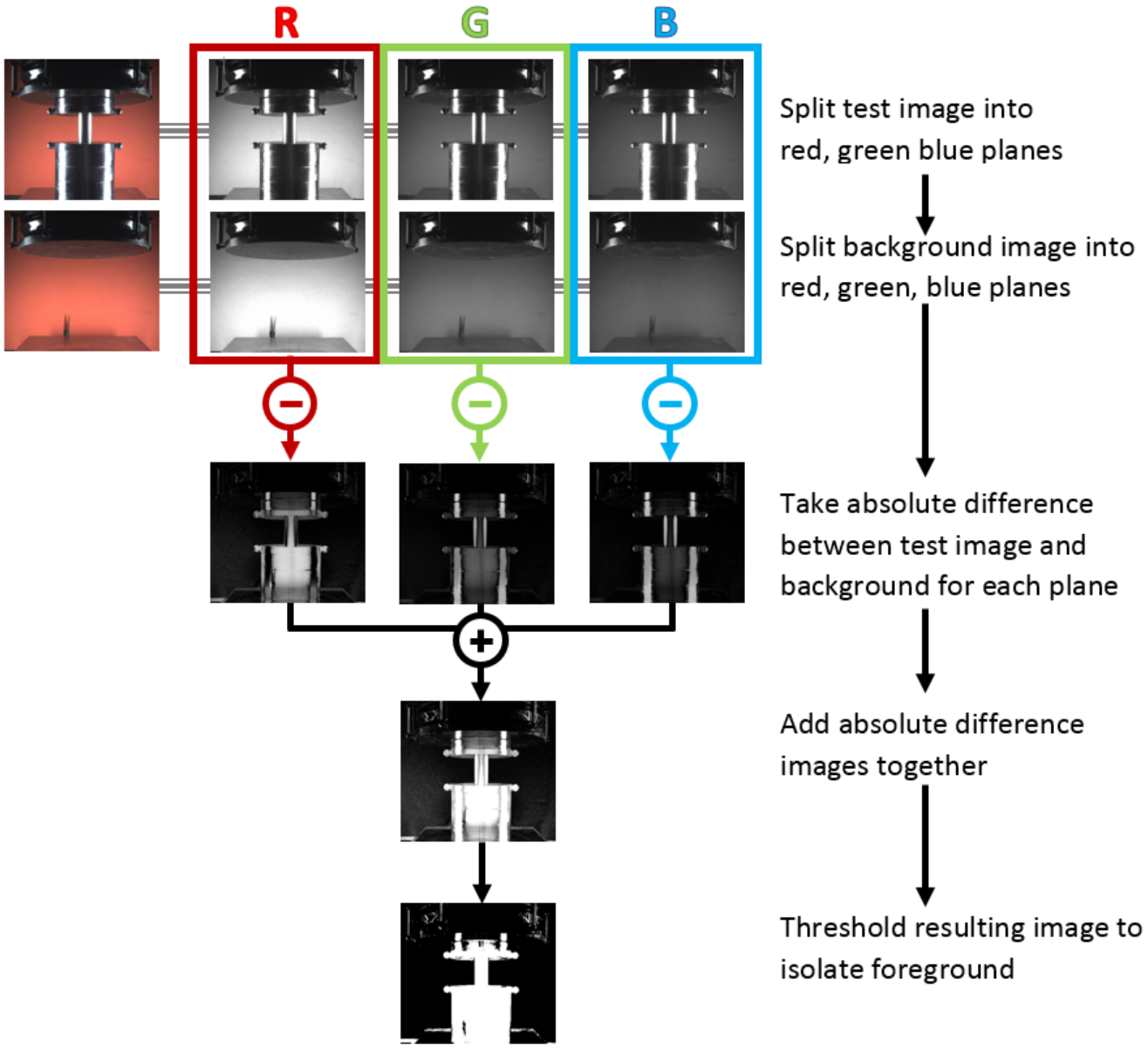


Figure 3.9: The background subtraction algorithm

3.4.3 LabVIEW: Edge Detection

A rake algorithm is used on the background-subtracted binary image to detect the edges of the test sample. In the context of image analysis, a rake algorithm is an edge detection algorithm which finds edges along a series of parallel search lines. In SPECS's implementation of the algorithm, the search lines are simply straight horizontal rows of pixels.

In a grayscale or binary image, an edge is detected whenever a sufficiently sharp jump or fall in pixel intensity is encountered ("sufficiently sharp" determined through user specified values of edge width and height). This is shown schematically in Figure 3.10 for a single search line on a simple silhouette image.

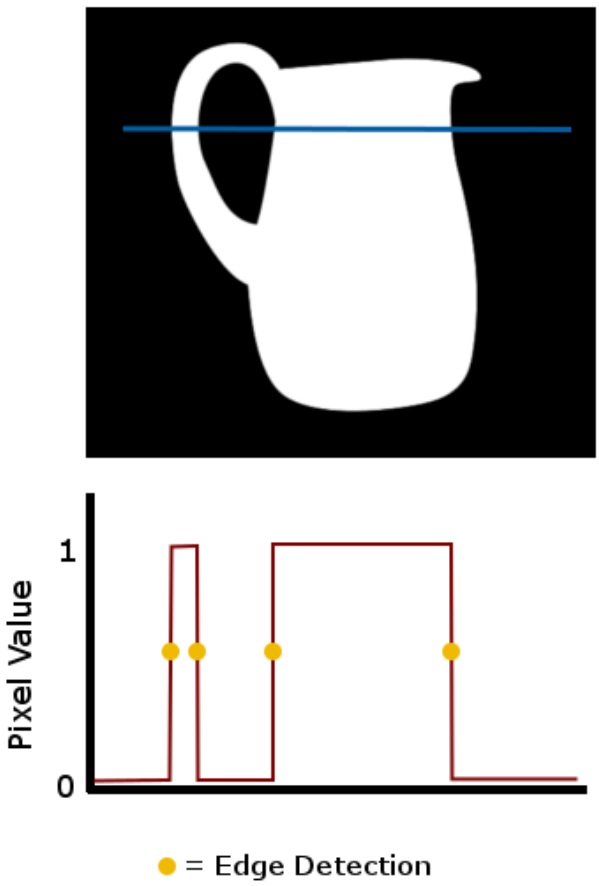
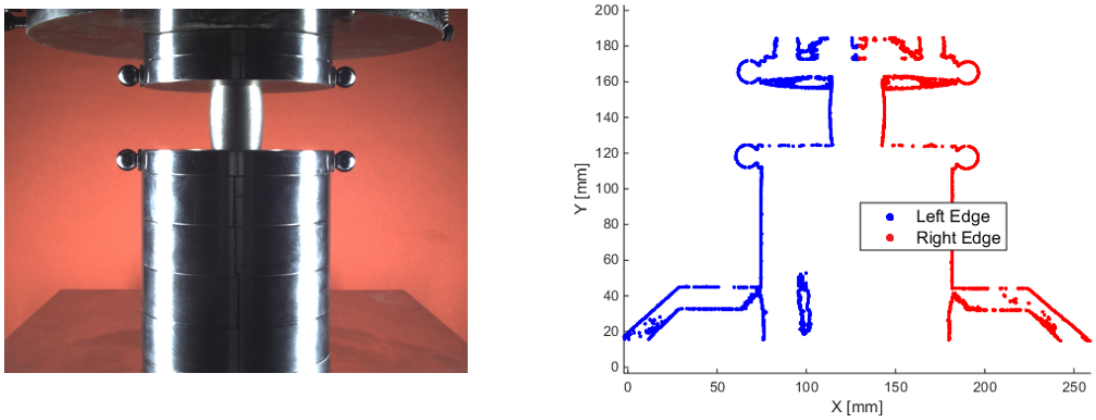


Figure 3.10: The rake edge detection algorithm

This process is repeated across evenly spaced search lines which cover the entire region of interest. This produces a cloud of (x,y) coordinates, like those seen in Figure 3.11.



(a) Original camera image

(b) Detected edges

Figure 3.11: Edge detection example

3.4.4 LabVIEW: Circle Detection

Spherical tooling balls inserted into the upper and lower platens serve as distinct visual markers to track sample height. When viewed in a 2D image, the spheres appear as circles. By determining the position of these circles, the position of the platens themselves may also be determined. This is done using a shape-matching algorithm built into LabVIEW's vision acquisition package. The shape-matching algorithm detects edges in an image and compares them to a template of a perfect circle with user specified parameters for radius and maximum acceptable occlusion. The edge is assigned a "match score", based on how well the edge conforms to the specified shape. The circle center coordinates, radius and match score are output to a data file, which is later filtered by radius and match score to isolate just the circles which correspond to the platen markers.

3.4.5 MATLAB: Data Synchronization

The data used as input for the MATLAB software does not have a common variable to which the data sets can all be referenced. Strobe and FSR voltage, originating from the data acquisition software, are referenced to the internal clock on the DAQ chassis. Load data, originating from the Instron press, are referenced to the Instron's internal clock. Edge and circle data, originating from the LabVIEW postprocessing script, are referenced to the frame numbers of the images. The first step in the

MATLAB software is to sync all of these dissimilar data sets to a common reference variable.

Frame numbers are taken as the common reference variable to which all datasets can be synced. Each camera frame has a distinct start and end time, with image exposure covering the entire time in between. Recall that the strobe line is a digital line that is set HIGH when the camera is on standby, and LOW when the camera is exposing an image. As seen in Figure 3.12, the start and end timestamps for each frame are found simply by extracting the falling and rising edges of the strobe line data, respectively.

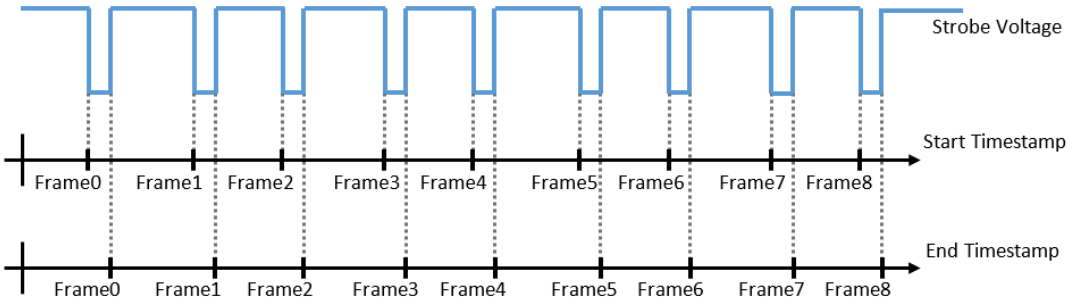


Figure 3.12: Detecting start and end timestamps from strobe line

The voltage signal from the force sensing resistor (FSR) is used to sync loading data from the Instron press to the camera frames. The FSR is placed in the bottom leg of a voltage divider, like the one shown in Figure 3.13. A fixed resistor is placed on the top leg, sized to be in the middle of the FSR's effective range. FSRs vary their resistance by several orders of magnitude when pressed, so the voltage measured at V_{out} tends towards zero when the FSR is at rest ($R_{FSR} >> R_{fixed}$), and tends towards V_{ref} when the FSR is pressed ($R_{FSR} << R_{fixed}$). The resulting plot of V_{out} mimics a plot of the force applied to the FSR, with a button press registering as a peak in voltage, as seen in Figure 3.14.

The strobe line and FSR voltage line are timed by the same clock on board the DAQ chassis, and are therefore already synchronized with each other. The timestamp of the peak FSR voltage is subtracted from every timestamp in the FSR and strobe datasets, which effectively shifts the timestamps so that FSR voltage peaks at $t = 0$ s. Physically, this peak occurs when the Instron frame's start button is pressed, which

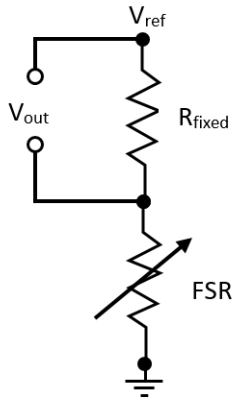


Figure 3.13: Voltage divider circuit for observing FSR

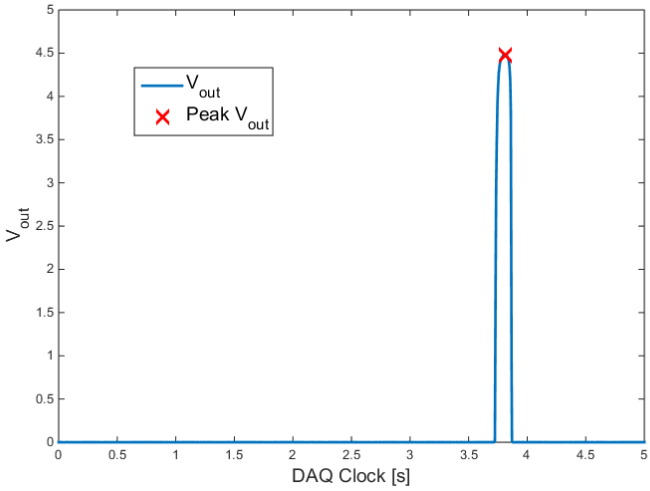


Figure 3.14: Typical signal from FSR voltage divider

coincides with $t = 0$ s as recorded by the Instron's own internal clock. As a result of this shifting operation, the start and end timestamps for each image align with the timestamps on the Instron load data.

The final data alignment step is to assign a single load value to each image. Load data which falls between the start and end timestamps of a single image is averaged, and assigned the same frame number. The timescale over which the averaging occurs is short, with a single image exposure taking between 330 and 340 ms. During testing, load ramps linearly over a period of more than six minutes, which means that the load changes by less than 0.1% of the total load range during a single image exposure.

3.4.6 MATLAB: Height and Diameter Calculation

With data syncing completed, each image has an associated frame number, along with circle data, edge data, and load data. To begin calculating stress and strain, the circle and edge data must first be converted into height and diameter data.

Sample height is calculated from the positioning of the four spherical platen markers. The center of each marker is in line with the center of the associated platen. For each platen, the heights of the two spherical markers are averaged, and the result is offset by a distance equal to half the platen thickness to calculate the position where the platen contacts the test sample.

Next, edge data is filtered to extract the sample's edges from the rest of the surrounding equipment. Any edge coordinate to the left of the centerline is classified as a left edge, and any coordinate to the right of the centerline is classified as a right edge. Next, the platen heights calculated in the previous step are used to set a vertical region of interest. A static offset is applied to the platen heights to ensures that tooling does not factor into diameter calculations. This results in a small loss of edge information at the very top and bottom of the sample.

Finally, diameter is calculated within the region of interest. First, the region of interest is subdivided into many small bands which stretch across the entire region of interest horizontally, and a few pixels each vertically, as shown in Figure 3.15. The distance between the average right coordinate and average left coordinate is taken as the diameter for that band, as shown in Figure 3.16.

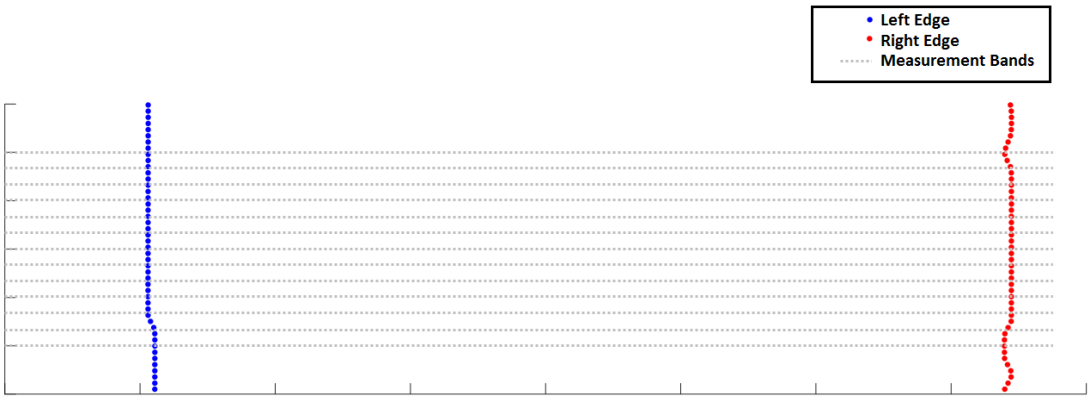


Figure 3.15: Edge data divided into bands for diameter measurement

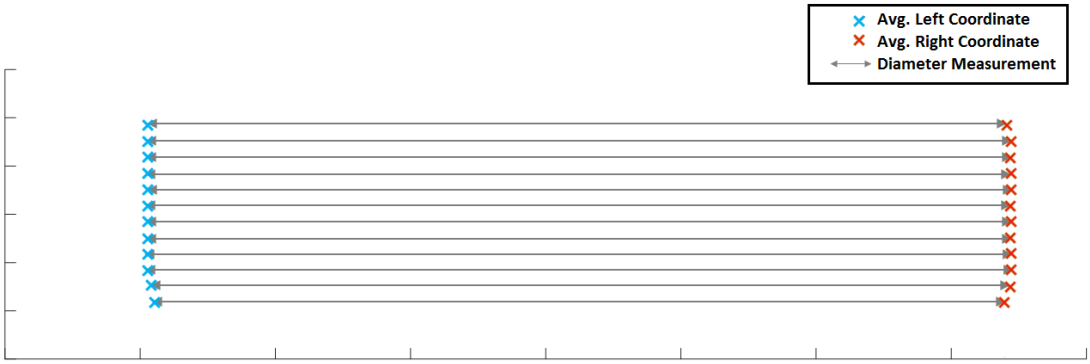


Figure 3.16: Averaged edge data for diameter measurement

3.4.7 MATLAB: Stress-Strain Calculation

In the final processing step, the MATLAB software combines the load, height and diameter data for each image to calculate stress and strain. For each image, both engineering stress/strain and true stress/strain are calculated. Engineering stress and strain are calculated using Equations 3.1 and 3.2 below, and consider only the initial sample dimensions in their calculations. True stress and strain, calculated using Equations 3.3 and 3.4, consider the instantaneous sample dimensions.

$$\sigma_{eng} = \frac{P}{A_o} \tag{3.1}$$

$$\epsilon_{eng} = \frac{L_i - L_o}{L_o} \tag{3.2}$$

where:

σ_{eng} = engineering stress

ϵ_{eng} = engineering strain

P = compressive force

A_o = original cross-sectional area of undeformed sample

L_o = original undeformed length of sample

L_i = instantaneous deformed length of sample

$$\sigma_{true} = \frac{P}{A_i} \tag{3.3}$$

$$\epsilon_{true} = ln\frac{L_i}{L_o} \tag{3.4}$$

where:

σ_{true} = true stress

ϵ_{true} = true strain

P = applied load

A_i = instantaneous cross-sectional area of deformed sample

L_o = original undeformed length of sample

L_i = instantaneous deformed length of sample

With the above formulas, compressive forces yield negative stresses and strains, while tensile forces yield positive stresses and strains. Throughout this thesis, compressive stresses and strains are referenced using absolute values, as is common when reporting results for compressive testing.

3.5 System Application and Limitations

As with any tool, SPECS has limitations which dictate how it can be used and how its results should be interpreted. Similar to the 2D extensometers reviewed in Section 2.1.2, SPECS is unable to account for sample movement out of plane, and any such movement will be falsely detected as the sample growing or shrinking. SPECS is silhouette based, and after accounting for perspective effects, the widest plane on the sample detects as its silhouette. Shapes which produce a silhouette that will not move out-of-plane include cylindrical samples and thin plates loaded in plane stress. In addition to sample shape, the aspect ratio and loading of a test sample must be selected to prevent buckling, which would lead to out-of-plane motion.

Caution must be taken when using SPECS with anisotropic materials. When testing cylindrical samples with SPECS, it is assumed that sample deformation will

take place axisymmetrically. As long as deformation is axisymmetric, the sample's silhouette will correspond to its mid-plane geometry, and this mid-plane will not move out-of-plane with the camera. If the sample is oriented in such a way that it deforms asymetrically, this may no longer be true.

When interpreting SPECS results, the degree of barreling in the sample must be considered. When friction is present, as in a cold upsetting test, the stress field within the test sample is non-uniform. Furthermore, the diameter of the sample varies throughout the barreled profile. True stress can be approximated using minimum, mean, or maximum diameters from SPECS data, but at high strains, these numbers will begin to diverge.

Chapter 4: Calibration Experiments

4.1 Height Calibration

4.1.1 Methods

A height calibration test used to assess how accurately SPECS is able to track platen height via the upper and lower platen markers. A stack of 4140 platens, as shown in Figure 4.1 were placed in the Instron. The marked upper and lower platens were placed at the top of the stack, separated by a plain 1" thick platen. A Starrett 3752 height gauge, shown in Figure 4.2 was used throughout the experiment to measure the height of the Instron piston.

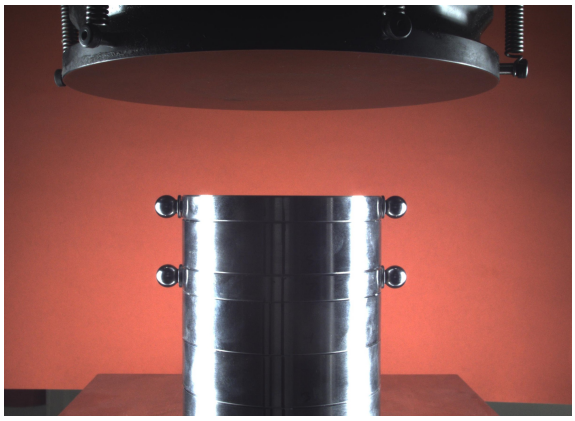


Figure 4.1: Platen setup for height calibration tests

Figure 4.2: Starrett 3752 height gauge

The Instron's height measurement and the height gauge were both zeroed at the start of the experiments. The Instron piston was raised by 1 mm increments until

a height of 20 mm, then by 2 mm increments to the piston's maximum possible height of 66 mm. At each increment, SPECS gathered data for approximately 10 frames. The Instron's height readout was recorded, as well as three height gauge measurements.

4.1.2 Results

Figure 4.3 below shows the height measurements made by SPECS and the Instron plotted against measurements made by the height gauge. The SPECS vs Height Gauge and Instron vs Height Gauge datasets deviate from each other slightly between nominal displacements of about 0 - 20 mm, but otherwise the data sets appear to agree quite well with each other.

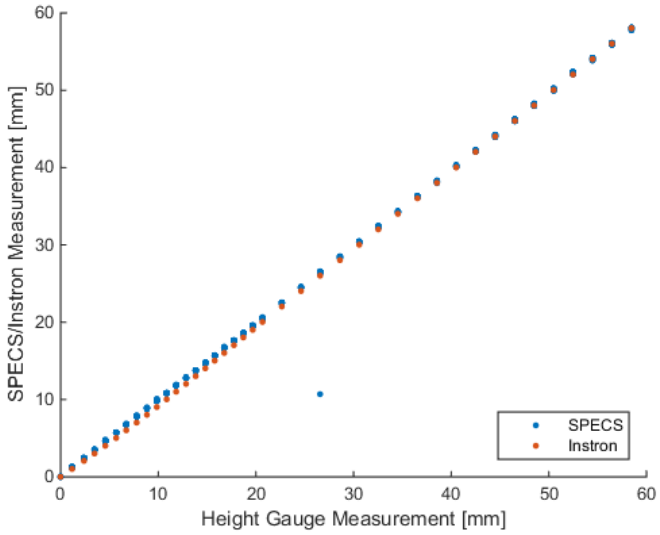


Figure 4.3: SPECS/Instron Measurements Vs Height Gauge

Figure 4.4 takes the average height gauge measurement to be the ground-truth for each data point. The SPECS measurement error and Instron measurement error are calculated by taking the difference between the Instron/SPECS measurement and the height gauge measurement.

From Figure 4.4, it is clear that both SPECS and the Instron undershoot the measurement given by the height gauge, though it should be noted that for both datasets, the error is within ± 1 mm.

Interestingly, the Instron error is nonlinear, with the most severe error occurring

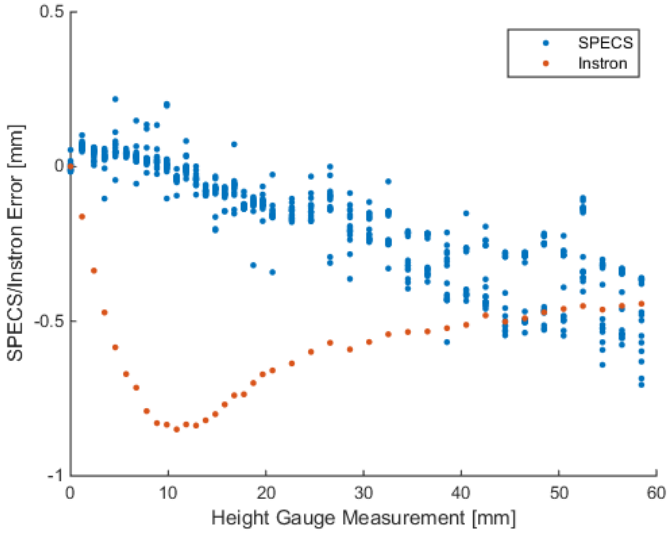


Figure 4.4: SPECS/Instron Errors Vs Height Gauge

in the first half of the dataset. It is tempting for researchers to take the values reported by their test frame as a ground-truth measurement, as the positional data is automatically logged and synced with loading data through a professionally designed and calibrated machine. However these results suggest that at the low loads and displacements used for our calibration experiments, the Instron's displacement results may not be the best dataset for comparison compared to a dedicated height gauge.

The SPECS measurement error, on the other hand, is quite linear. From this graph, it is clear that the SPECS measurements drift from the height gauge measurements, leading to error that is more severe at larger displacements.

Figures 4.5 and 4.6 are examples of scatterplots used to compare two datasets which share a common variable. In the case of the height gauge experiments, all height measurements were taken simultaneously for a given nominal height, allowing the height gauge measurement (represented as the color axis) to be taken as the common variable for comparison. As data processing techniques are improved and refined, the old scatterplots can be compared to new ones to assess the improvements at a glance. Any improvement to either variable will show up as a reduction in the data's spread along that axis.

Figure 4.5 shows the SPECS measurement error for the top platen on the X axis, and the Instron measurement error on the Y axis. The dataset predominantly resides

in the bottom left quadrant, clearly showing the tendency of both measurement techniques to undershoot the height gauge measurement.

Figure 4.6 shows SPECSs measurement error on both X and Y axes, with the error for the top platen on the X axis, and error for the bottom platen on the Y axis. The drift between SPECS and height measurements is apparent from the data's movement away from the center of the graph at high displacements. The graph also clearly shows that there is little bias in the SPECS measurement, with the top and bottom platens showing similar magnitudes of error throughout the dataset.

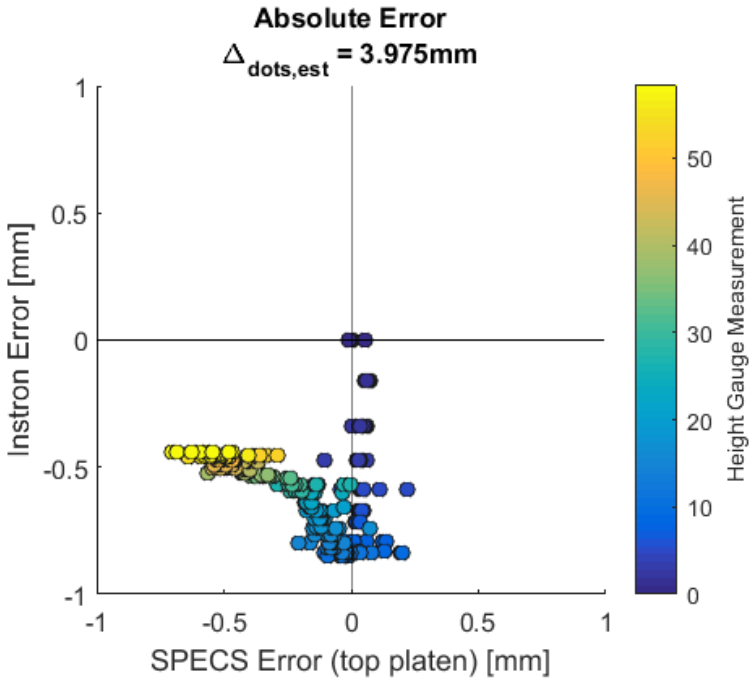


Figure 4.5: SPECS/Instron Errors Vs Height Gauge (Scatter Plot)

The consistent under-shooting of measurements by SPECS was theorized to be a result of the calibration procedure. As described in Chapter 3, the calibration procedure obtains real-world coordinates for each pixel in an image by comparing to a calibration image consisting of a grid of dots, spaced a known distance apart. For initial testing, this distance was estimated by using a ruler with 1 mm markings to measure the distance between the dots at the start and end of a row, and dividing by the number of dots in between. This initial estimate was found to be $\Delta_{dots,est} = 3.975$ mm. It can be shown that if this initial estimate differs from the

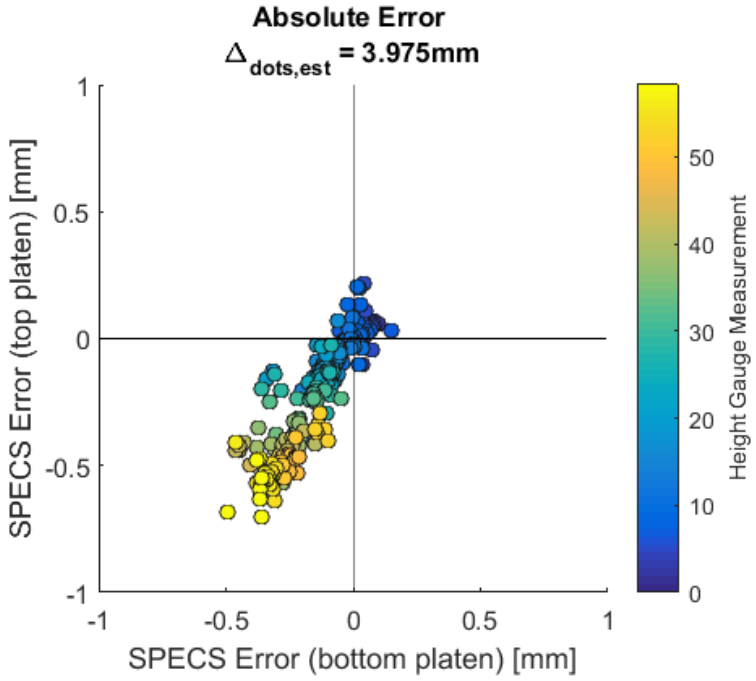


Figure 4.6: SPECS Errors Vs Height Gauge (Scatter Plot)

true spacing such that $\Delta_{dots,est} = C_{dots}\Delta_{dots,true}$ the resulting diameter measurements differ by the same factor:

Given:

$$\Delta_{dots} = \text{Distance between calibration dots [mm]}$$
$$X_{i,mm} = \text{X coordinate of edge i [mm]}$$
$$X_{i,px} = \text{X coordinate of edge i [px]}$$
$$C_{dots} = \text{Factor by which nominal dot spacing has been incorrectly estimated}$$

Consider Figure 4.7, which is a schematic representation of a typical calibrated image. The top-left most pixel is taken as (0,0) in both the real-world and pixel coordinate systems. The diameter of the sample ($D_{sample,true}$) along a given line can be calculated from the two edge detections (X_1, Y_1) and (X_2, Y_2):

$$D_{sample,true} = X_{2,mm} - X_{1,mm}$$
$$D_{sample,true} = \Delta_{dots,true}(X_{2,px} - X_{1,px}) \qquad \Longleftarrow \text{substitute } X_{i,mm} = \Delta_{dots} \times X_{i,px}$$

Considering the case where an underestimated $\Delta_{dots,est}$ is used:

$$
\begin{aligned}
D_{sample,est} &= \Delta_{dots,est}(X_{2,px} - X_{1,px}) \\
D_{sample,est} &= C_{dots}\Delta_{dots,true}(X_{2,px} - X_{1,px}) \\
&\therefore \\
D_{sample,est} &= C_{dots}D_{sample,true}
\end{aligned} \tag{4.1}
$$

From the above proof, a critical result was realized: a better estimate of Δ_{dots} could be obtained by analyzing calibrated measurement data. Multiplying the output data by a correction factor would have the same effect on results as multiplying Δ_{dots} by the same correction factor prior to data processing. It follows, then, that optimization techniques applied to the output data could be used to obtain an inter-dot spacing correction factor, f_{dots}, which could be applied before or after data processing to correct for the coarse measurement techniques used to initially find Δ_{dots}.

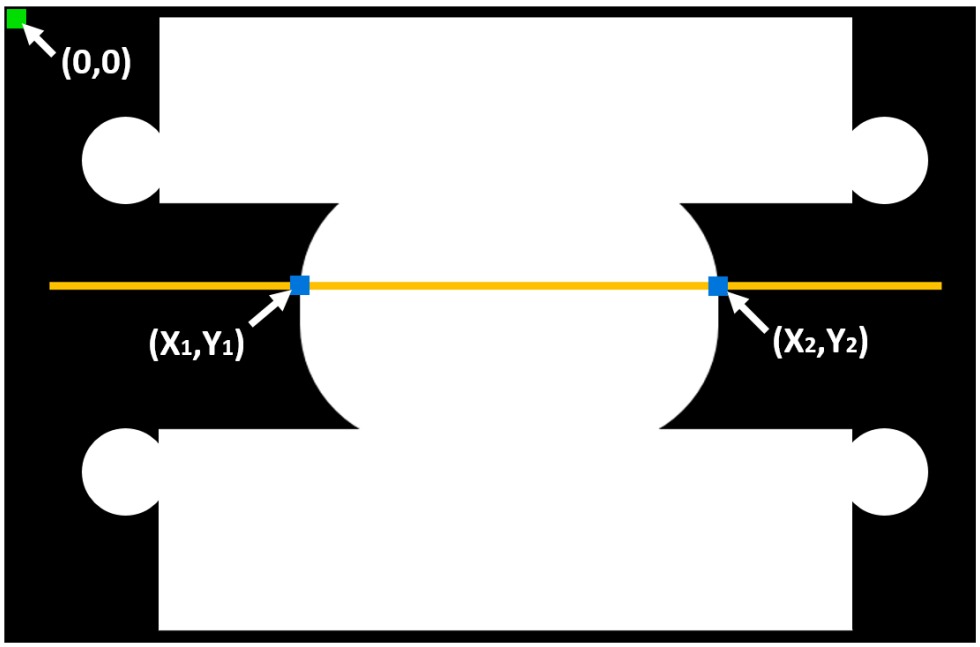


Figure 4.7: Proof setup: effect of C_{dots} on diameter measurement

The method of linear least squares, a common method for fitting data which estimates parameters in a linear equation by minimizing the summed square of the data's residuals, served as a starting point for optimizing f_{dots}. In this method [31],

data is fit to a model of the form :

$$y = X\beta + e \tag{4.2}$$

in matrix form:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,m} \\ X_{2,1} & X_{2,2} & \dots & X_{2,m} \\ \vdots & \vdots & & \vdots \\ X_{n,1} & X_{n,2} & \dots & X_{n,m} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} \tag{4.3}$$

where

y_i = independent variable at data point i

$X_{i,j}$ = dependent variable j at data point i

β = matrix containing the m coefficients to be fit

e_i = error on measurement i

An optimal, closed form solution for β that minimizes the error term e is given as:

$$\beta = (X^T X)^{-1} X^T y \tag{4.4}$$

Data from the height calibration experiments was re-arranged to match the format seen in Equation 4.2:

$$\delta_{HG} = \delta_S f_{dots} + e_{SPECS} \tag{4.5}$$

Where

δ_{HG} = platen displacement measured by height gauge

δ_S = platen displacement measured by SPECS

f_{dots} = inter-dot spacing correction factor

e_S = error between SPECS measurements and height gauge measurement

The inter-dot spacing correction factor can then be found as:

$$f_{dots} = (\delta_S^T \delta_S)^{-1} \delta_S^T \delta_{HG} \tag{4.6}$$

Which can be applied to the original inter-dot spacing to arrive at a better estimate for calibration:

$$\Delta_{dots,cor} = f_{dots} \Delta_{dots,est} \tag{4.7}$$

Applying Equations 4.6 and 4.7 to height calibration data yields an optimal correction factor $f_{dots} = 1.0074$ and a corrected inter-dot spacing of $\Delta_{dots,cor} = 4.004$ mm.

Figures 4.8 and 4.9 show Instron and SPECS error as a function of nominal height gauge measurement. Figure 4.8 is processed using an the original inter-dot spacing of $\Delta_{dots,est} = 3.975$ mm for calibration. Figure 4.9a applies the corrected value by multiplying the original processed data by f_{dots}, whereas Figure 4.9b shows data that has been reprocessed from raw images using $\Delta_{dots,cor} = 4.004$ mm as the inter-dot spacing for calibration. As expected, Figures 4.9a and 4.9b are indistinguishable from eachother, supporting the assumptions used to derive equations 4.6 and 4.7. The graphs in Figure 4.9 also show marked improvement in over Figure 4.8, with SPECS error contained to a much narrower range when $\Delta_{dots,cor} = 4.004$ mm is used.

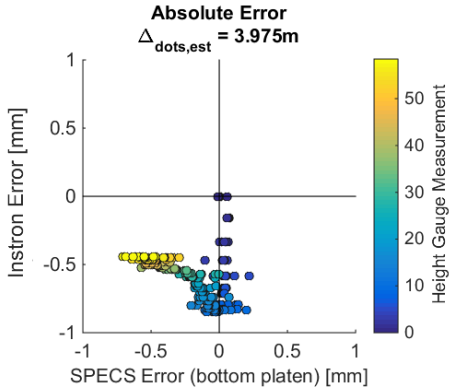
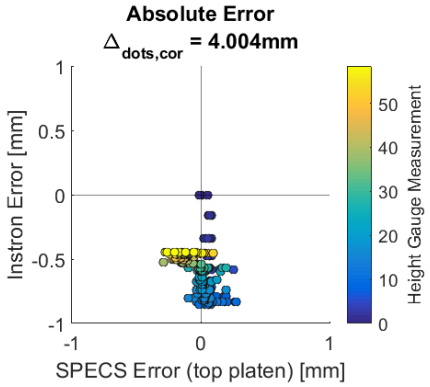
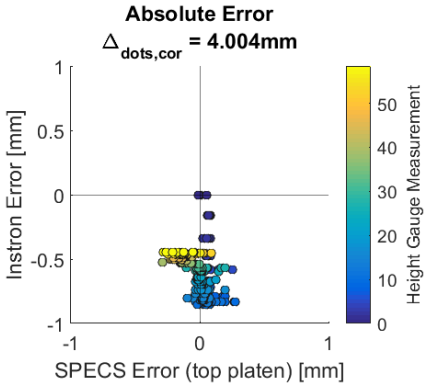


Figure 4.8: Instron/SPECS error vs Height Gauge, $\Delta_{dots} = 3.975$ mm

The improvement to SPECS error through use of the corrected $\Delta_{dots,cor}$ is made more clear in Figure 4.10, which plots SPECS error for the upper and lower platen versus height gauge measurment. Figure 4.10a, which uses $\Delta_{dots,est} = 3.975$ mm,



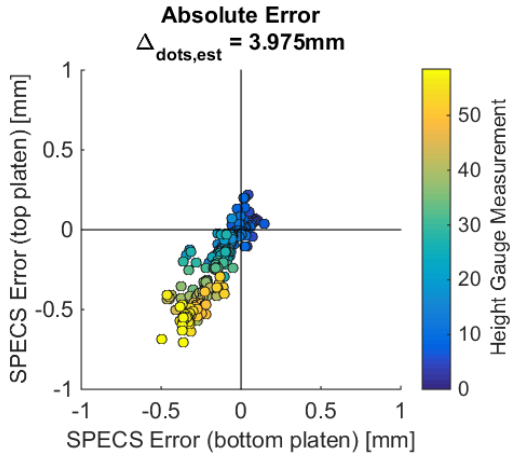
(a) Instron/SPECS error vs Height Gauge - Dot spacing corrected after postprocessing



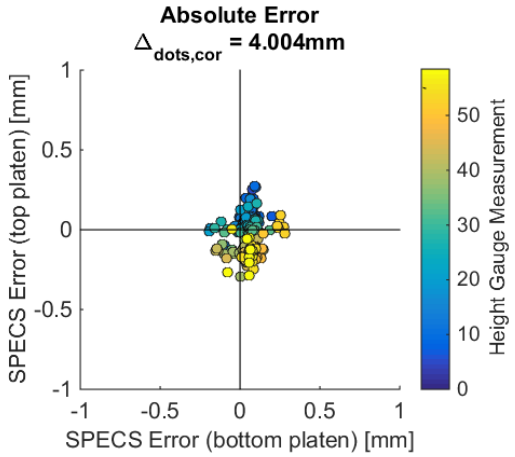
(b) Instron/SPECS error vs Height Gauge - Dot spacing corrected before calibration

Figure 4.9: Instron/SPECS error vs Height Gauge, $\Delta_{dots} = 4.004$ mm

the drift between SPECS measurements and height gauge measurements is clear. Using the corrected $\Delta_{dots,cor} = 4.004$ mm as in Figure 4.10b, the drift is eliminated, and the error is seen as a random cloud around (0,0).



(a) SPECS error vs Height Gauge $\Delta_{dots,est} = 3.975$ mm



(b) SPECS error vs Height Gauge $\Delta_{dots,cor} = 4.004$ mm

Figure 4.10: SPECS error vs Height Gauge, before and after correcting Δ_{dots}

4.2 Calibration Sheet Microscopy

The results of the height calibration experiments strongly suggested that there were inaccuracies in the original measurement of the spacing between calibration dots. To verify this hypothesis, the calibration sheet was examined under an optical microscope.

4.2.1 Methods

Figure 4.11 shows a schematic of the SPECS calibration sheet, with the numbering convention for the rows and columns shown. Microscope images were saved using image capture software for nine rows and nine columns of calibration dots. Images were batch processed using ImageJ image analysis software. A flowchart for the image processing macro can be seen in Figure 4.12.

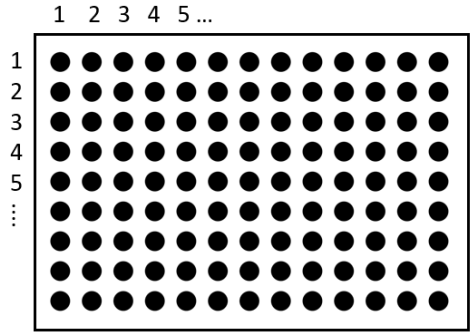


Figure 4.11: SPECS calibration sheet with numbered columns/rows

ImageJ particle measurement results were compiled for 9 rows to calculate horizontal dot spacing (Δx) and for 9 columns to calculate vertical dot spacing (Δy).

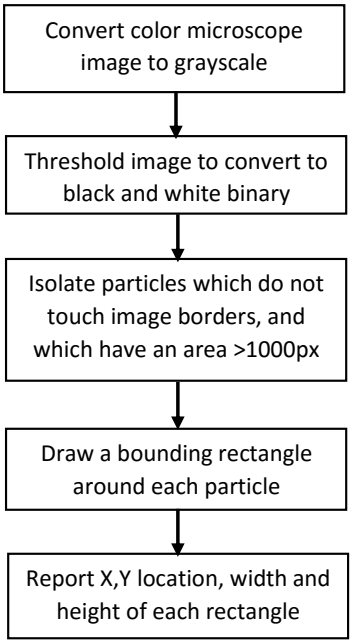


Figure 4.12: ImageJ macro for processing microscope images

4.2.2 Results

At the magnification used to measure the calibration sheet, images from the optical microscope have a resolution between 84 and 88 px/mm. Using an average value of 86 px/mm, the distance between bounding rectangle centroids was calculated in mm. Figure 4.13 shows the results for Δx and Figure 4.14 shows the results for Δy. The results for each row and column can be found in Tables 4.1 and 4.2.

It can be seen in the graph for Δy (Figure 4.14) that vertical spacing is not perfectly consistent throughout the calibration sheet. Rather, the dots closer to the bottom of the sheet appear to be slightly closer together than those at the top of the sheet. Furthermore, measured dot spacing fell over a relatively large range of 3.860 to 4.081 mm.

The variation in calibration dot spacing is not entirely unexpected, given how the calibration sheet was manufactured. In its current iteration, the calibration sheet is printed on a consumer-grade laser printer, and glued to a stiff metal backing. For future work, it is recommended that higher accuracy methods are used to produce the calibration sheet, such as printing on a high accuracy plotter engraving directly

on a metal sheet using a laser engraver.

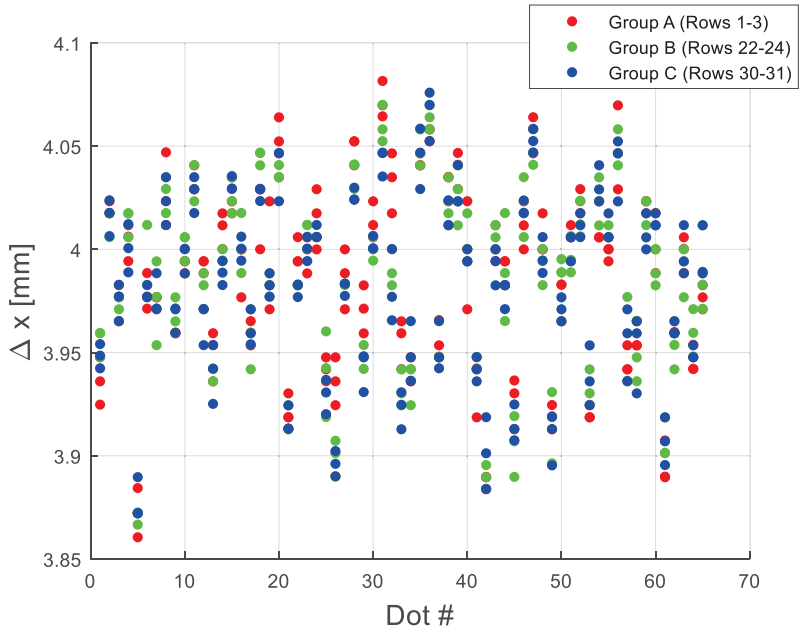


Figure 4.13: Calibration sheet microscopy, Δx

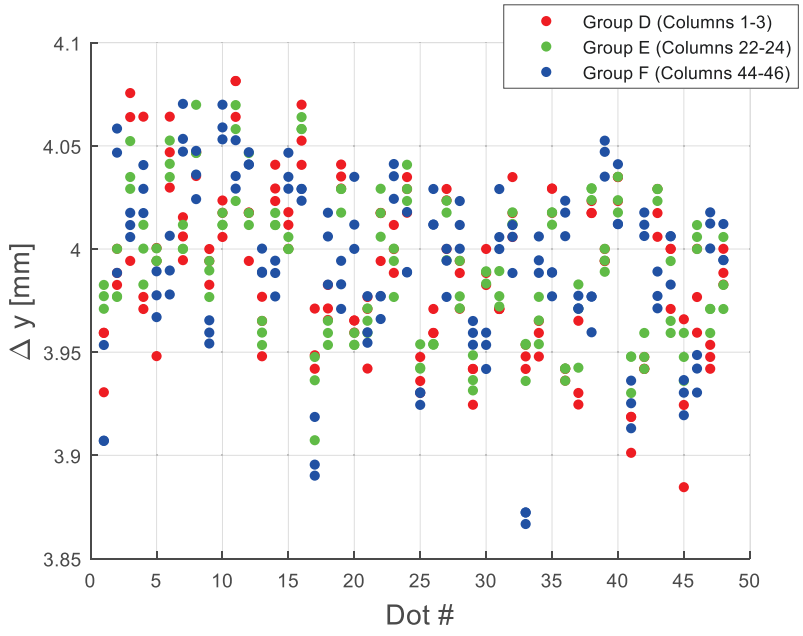


Figure 4.14: Calibration sheet microscopy, Δy

Row Number	Δx_{min} [mm]	Δx_{max} [mm]	Δx_{mean} [mm]
Row 1	3.8605	4.0698	3.9846
Row 2	3.8721	4.0698	3.9896
Row 3	3.8838	4.0816	3.9861
Row 22	3.8666	4.0640	3.9823
Row 23	3.8724	4.0698	3.9873
Row 24	3.8721	4.0700	3.9834
Row 30	3.8721	4.0759	3.9806
Row 31	3.8896	4.0700	3.9854
Row 32	3.8724	4.0585	3.9814

Table 4.1: Row-by-row horizontal calibration dot spacing (Δx)

Column Number	Δy_{min} [mm]	Δy_{max} [mm]	Δy_{mean} [mm]
Column 1	3.8846	4.0815	3.9873
Column 2	3.9186	4.0756	3.9929
Column 3	3.9186	4.0814	3.9903
Column 21	3.9361	4.0698	3.9898
Column 22	3.9302	4.0699	3.9937
Column 23	3.9073	4.0582	3.9899
Column 44	3.8723	4.0700	3.9917
Column 45	3.8723	4.0704	3.9950
Column 46	3.8667	4.0534	3.9895

Table 4.2: Column-by-column vertical calibration dot spacing (Δy)

From Tables 4.1 and 4.2, a mean calibration dot spacing can be calculated horizontally and vertically as $\Delta x = 3.9845$ mm and $\Delta y = 3.9911$ mm respectively. These values are used for all other experiments as the most accurate measurements available.

Considering the range of Δx and Δy values measured, there is a notable degree of variation on calibration dot spacing. For horizontal spacing, values ranged from $\Delta x_{min} = 3.8605$ to Δx_{max} 4.0816 mm, or -3.1% - +2.4% of the nominal value. For vertical spacing, values ranged from $\Delta y_{min} = 3.8667$ to Δy_{max} 4.0815 mm, or -2.6% - +2.3% of the nominal value.

4.3 Calibration Blocks (Width Calibration)

4.3.1 Methods

A set of four aluminum calibration blocks were produced through CNC machining to provide an accurate standard to compare SPECS measurements to. Two straight-edged and two curved samples were produced, the major dimensions of which are labelled in Figure 4.15 below.

The calibration blocks were placed in the Instron press as though preparing for a cold upsetting test, with platens of 4140 steel placed above and below the sample. The SPECS system was run to collect at least 20 frames of image data, which was processed using SPECS postprocessing software. As the Instron was not run during data collection to avoid placing load on the calibration samples, placeholder load data was constructed to allow the postprocessing code to function as normal. This placeholder data was deemed acceptable, as no load-dependent calculations were performed on any of the calibration block data.

20 frames were analyzed for each calibration block. For blocks 1 and 3 (straight edge blocks), the maximum, minimum and mean diameters, as well as the standard deviation on diameter were analyzed for each frame. For blocks 2 and 4 (curved edge blocks), just the maximum diameters were analyzed for each frame.

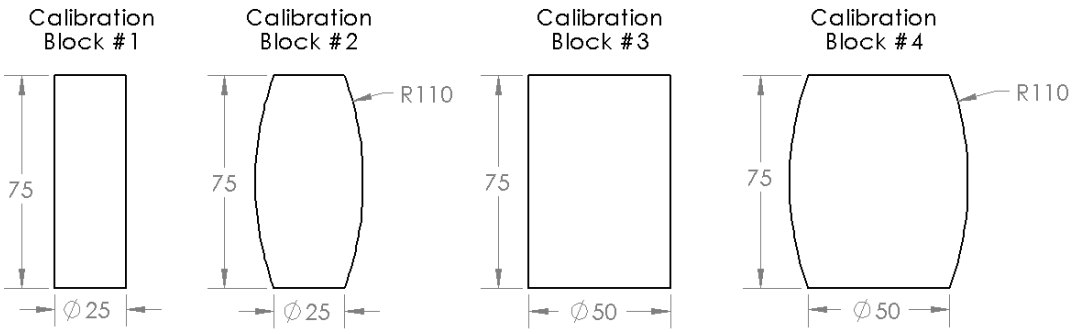


Figure 4.15: Calibration blocks (dimensions in mm)

4.3.2 Results

For each calibration block, 20 frames worth of data were analyzed. Each frame was considered in isolation for calculating the max, and mean diameter of the frame. A standard deviation was calculated for between the 20 individual frame values to give a measure of how much a measurement could differ from frame to frame.

Table 4.3 shows the statistics calculated for maximum diameter, D_{max}, along with the true measured value of D_{max} for each calibration block. All values calculated for D_{max} undershot the true, measured diameter. The errors seen in the 20 frame average D_{max} are shown in Table 4.4.

Calibration Block	True D_{max} [mm]	20 Frame Avg D_{max} [mm]	20 Frame Range of D_{max} [mm]	Standard Deviation [mm]
Block 1	25.13	24.77	24.68 - 24.83	0.0406
Block 2	38.18	37.60	37.53 - 37.74	0.0518
Block 3	50.07	49.44	49.36 - 49.49	0.0420
Block 4	63.19	62.31	62.26 - 62.52	0.0563

Table 4.3: Calibration block statistics: maximum diameter

Calibration Block	True D_{max} [mm]	20 Frame Avg D_{max} [mm]	Absolute Error [mm]	Relative Error
Block 1	25.13	24.77	0.36	-1.4%
Block 2	38.18	37.60	0.58	-1.5%
Block 3	50.07	49.44	0.63	-1.3%
Block 4	63.19	62.31	0.88	-1.4%

Table 4.4: Calibration block statistics: error on maximum diameter

Table 4.5 shows the statistics calculated for minimum diameter, D_{min}, along with the true measured value of D_{min} for each calibration block. All values calculated for D_{min} undershot the true, measured diameter. Because of this undershoot, the errors on D_{min} are larger than those on D_{max}. The errors seen in the 20 frame average D_{min} are shown in Table 4.6.

Calibration Block	True D_{min} [mm]	20 Frame Avg D_{min} [mm]	20 Frame Range of D_{min} [mm]	Standard Deviation [mm]
Block 1	25.05	24.34	24.30 - 24.38	0.0233
Block 3	50.05	48.60	48.37 - 48.80	0.0852

Table 4.5: Calibration block statistics: minimum diameter

Calibration Block	True D_{min} [mm]	20 Frame Avg D_{min} [mm]	Absolute Error [mm]	Relative Error
Block 1	25.05	24.34	0.79	-3.2%
Block 3	50.05	48.60	1.47	-2.9%

Table 4.6: Calibration block statistics: error on minimum diameter

Table 4.7 shows the statistics calculated for mean diameter, D_{mean}, along with the true measured value of D_{mean}. As could be expected, the error on the average D_{mean} shown in Table 4.8 falls between the errors for D_{max} and D_{min}.

Calibration Block	True D_{mean} [mm]	20 Frame Avg D_{mean} [mm]	20 Frame Range of D_{mean} [mm]	Standard Deviation [mm]
Block 1	25.09	24.56	24.54 - 24.57	0.0066
Block 3	50.06	49.07	49.06 - 49.09	0.0077

Table 4.7: Calibration block statistics: mean diameter

Calibration Block	True D_{mean} [mm]	20 Frame Avg D_{max} [mm]	Absolute Error [mm]	Relative Error
Block 1	25.09	24.56	0.42	-1.7%
Block 3	50.06	49.07	0.99	-2.0%

Table 4.8: Calibration block statistics: error on mean diameter

Finally, the standard deviation of diameters measured *within* each frame was calculated for the two straight edged samples (1 and 3), the results of which are detailed in Table 4.9.

Calibration Block	Lowest σ_{frame} [mm]	Highest σ_{frame} [mm]	Average σ_{frame} [mm]
Block 1	0.0869	0.1035	0.0959
Block 3	0.1361	0.1582	0.1481

Table 4.9: Calibration block statistics: framewise standard deviations

Taken together, Tables 4.3 through 4.8 show that the optical measurements taken by SPECS for maxiumum diameter are very self-consistent, with the worst standard deviation between frames measuring less than 0.054 mm. However, there is a systematic bias which causes consistently low measurements, on the order of 1.3 - 1.5% relative error for average maximum diameter, and 2.9 - 3.2% relative error for average minimum diameter. This suggests that there is still room for tuning the system for future work. One likely place to start is the threshold used to convert images from grayscale to binary during the background subtraction algorithm. Lowering the threshold would result in more pixels near the test sample to be counted as foreground, resulting in larger measured diameters.

The results in Table 4.9 show promising consistency of measurement on the straight edged calibration blocks. Block 1 showed at worst $\sigma_{frame} = 0.1035$ mm while block 3 showed at worst $\sigma_{frame} = 0.1582$ mm, or about 0.4% and 0.3% of the nominal sample diameter, respectively.

Measurement error for the straight edged calibration blocks was plotted as a function of vertical position, to check for any bias in the measurement error. As seen in Figures 4.16 and 4.17, relative error peaked at just under -3.4%, showing a slight linear bias. Relative measurement error appears to be independent of sample diameter.

The microscopy experiments covered in Section 4.2 provide another possible explanation for the apparent offset and bias in the calibration block results. It was shown that inter-dot spacing varies throughout the calibration sheet by up to $\pm 3.1\%$ of the nominal spacings $\Delta x = 3.9845$ mm $\Delta y = 3.9911$ mm - this could explain part of the -3.4% error on diameter measurement.

Another possible explanation could be tilt within the calibration sheet itself. The calibration sheet is bolted to a magnetic base at its bottom, as shown in Figure 4.18, creating a cantilever with a free end at the top of the sheet. When measured using an inclinometer, the calibration sheet was found to be tilted at an angle of 2° when installed in the Instron press. At a height of 200 mm, this means that the top of the calibration sheet sits about 7 mm closer to the camera than the bottom.

An approximation for image magnification as a function of focal length and distance between object and lens is given in Equation 4.8 [32]. During testing, the

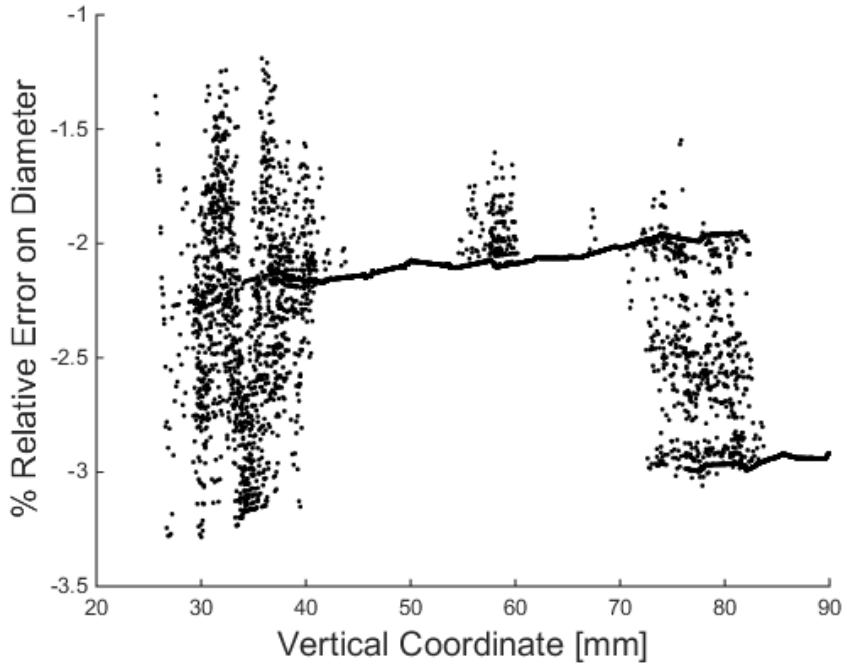


Figure 4.16: Relative error on diameter measurement, calibration block 1

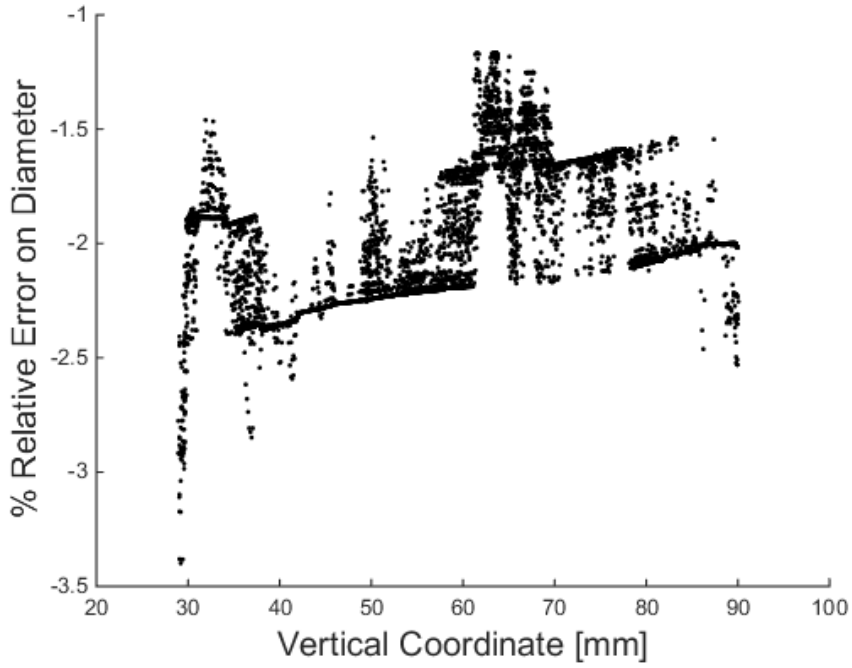


Figure 4.17: Relative error on diameter measurement, calibration block 3

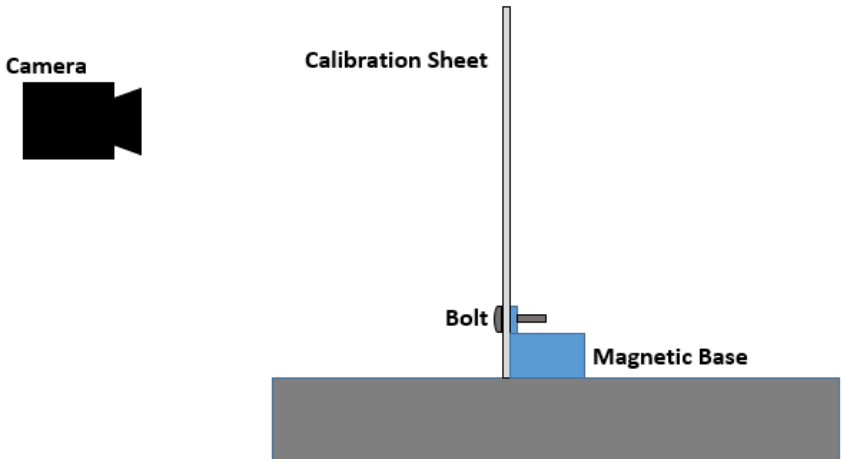


Figure 4.18: Schematic of calibration sheet - side view

camera sits approximately 450 mm from the calibration sheet, as measured from the bottom. Accounting for calibration sheet tilt, the ratio between magnification at the top of the calibration sheet and magnification at the bottom of the calibration sheet can be calculated, as seen in Equation 4.9.

$$M = \frac{f}{WD} \tag{4.8}$$

where:

$$\begin{aligned} M &= \text{magnification power} \\ f &= \text{focal length} \\ WD &= \text{working distance (distance between object and lens)} \end{aligned}$$

$$\begin{aligned} \frac{M_{top}}{M_{bottom}} &= \frac{f/WD_{top}}{f/WD_{bottom}} \\ &= \frac{WD_{bottom}}{WD_{top}} \end{aligned} \tag{4.9}$$

where:

$$\frac{M_{top}}{M_{bottom}} = \text{ratio of magnifications from top to bottom of calibration sheet}$$
$$WD_{top} = \text{working distance to top of calibration sheet}$$
$$WD_{bottom} = \text{working distance to bottom of calibration sheet}$$

Substituting 450 mm for WD_{bottom} and 443 mm for WD_{top}, Equation 4.9 yields a magnification ratio of 1.0158. In other words, with a tilted calibration sheet, dot spacing at the top of the sheet is falsely magnified by an estimated 1.6%. The equations used in the analysis are approximations suggested by Edmund Optics, and are recommended only for initial sizing estimates, not system calibration. However, these approximations do show the potential for significant error in calibration as a result of very small tilt angles on the calibration sheet.

4.4 Camera Resolution Analysis

When describing measurement errors, it is important to distinguish between *resolution* and *accuracy*. In the previous sections, the experiments dealt with accuracy: in other words, how well SPECS results measured the actual value of a dimension. Resolution, on the other hand, describes the minimum difference that can be detected between two measurements. For an optical system like SPECS, resolution errors result from the fact that digital images are stored as data organized as discrete pixels. The resolution of a single pixel, therefore, is the smallest measurement the system can resolve. A change in sample height or diameter will not produce a change in measurement until it is large enough to move the silhouette into a neighboring pixel.

Resolution was estimated using measurements made during the SPECS calibration process. As described in Chapter 3, the SPECS calibration process starts with the system comparing the pixel coordinates of detected reference points to their real-world coordinates calculated from dot spacing. These measurements were output as a text file, and analyzed using statistical tools in MATLAB. The results of this analysis can be seen in Table 4.10.

Average spacing between calibration dots was 31.08 px, for a real-world spacing of 3.9878 mm (an average value of Δx and Δy. This results in an average camera resolution of 0.128 mm/px. As suggested by the lens magnification equation (Equation 4.8), this resolution could be improved simply by moving the camera closer to

Direction	Δ_{min} [px]	Δ_{max} [px]	Δ_{mean} [px]	Standard Deviation [px]
Horizontal	29.94	32.74	31.02	.35
Vertical	30.33	32.82	31.13	0.28
Overall	29.94	32.82	31.08	0.32

Table 4.10: Calibration sheet spacing (in pixels)

the test sample, as a larger magnification means more pixels per millimeter. This may require a different camera mounting system, as the current system mounts the camera to the Instron press's door, a fixed distance from the test sample.

4.5 Overall Error Estimation

The calibration experiments conducted with SPECS reveal potential error for measuring platen position and sample diameter. These in turn will lead to inaccuracies in calculating stress and strain. Recall the equation for true strain:

$$\epsilon_{true} = ln\frac{L_i}{L_o} \tag{4.10}$$

where:

ϵ_{true} = true strain

L_o = original undeformed length of sample

L_i = instantaneous deformed length of sample

In this equation, L_o and L_i are calculated by subtracting the height of the bottom platen from the top platen. Both platen heights are subject to an absolute error of ± 0.5 mm. Propogation of errors on Equation 4.10 yields the following:

$$\begin{aligned} e_{\epsilon_{true}} &= \sqrt{\left(\frac{\partial \epsilon_{true}}{\partial L_i} e_{L_i}\right)^2 + \left(\frac{\partial \epsilon_{true}}{\partial L_o} e_{L_o}\right)^2} \\ &= \sqrt{\left(\frac{1}{L_i} e_{L_i}\right)^2 + \left(\frac{-1}{L_o} e_{L_o}\right)^2} \end{aligned} \tag{4.11}$$

$$e_{L_i} = e_{L_o} = \sqrt{(e_{platen,top})^2 + (e_{platen,bottom})^2}$$
$$= \sqrt{(0.5mm)^2 + (0.5mm)^2} = \frac{1}{\sqrt{2}}mm \tag{4.12}$$

where:

$e_{\epsilon_{true}}$ = error on true strain

e_{L_o} = error original undeformed length of sample

e_{L_i} = error instantaneous deformed length of sample

$e_{platen,top}$ = error measuring top platen position

$e_{platen,bottom}$ = error measuring bottom platen position

Combining Equations 4.11 and 4.12...

$$e_{\epsilon_{true}} = \sqrt{\left(\frac{1}{\sqrt{2}L_i}\right)^2 + \left(\frac{1}{\sqrt{2}L_o}\right)^2} \tag{4.13}$$

The formula for true stress is shown in Equation 4.14. There are two sources of error present when calculating true strain: measurement error on the load and measurement error on the diameter.

$$\sigma_{true} = \frac{P}{A_i} = \frac{P}{\frac{1}{4}\pi D_i^2} \tag{4.14}$$

where:

σ_{true} = true stress

P = applied load

A_i = instantaneous cross sectional area

D_i = instantaneous diameter

The Instron press has a nominal error of ±1% of the load reading [33], and SPECS has a diametral measurement error of −3.4%, as shown in Section 4.3. Propogation of errors on Equation 4.10 yields the following:

$$\begin{aligned} e_{\sigma_{true}} &= \sqrt{\left(e_{P,rel}\right)^2 + 2\left(e_{d,rel}\right)^2} \\ &= \sqrt{\left(0.01\right)^2 + 2\left(0.034\right)^2} \\ &= 0.0491 \\ &= 4.91\% \end{aligned} \tag{4.15}$$

where:

$e_{\sigma_{true}}$ = relative error on true stress

$e_{P,rel}$ = relative error on applied load

$e_{d,rel}$ = relative error instantaneous diameter

The calculated error on true stress is about 2% higher than results reported in the literature for similar cold upsetting tests. The ASTM standard test method for compression testing of metals [1], for example, reported precision error on yield stress of Al2024-T351 at 3.1% (for 95% repeatability within a single lab). Seidt and Gilat, with their comprehensive set of Al2024-T351 experiments reported that most of their stress errors fell below 3%. As error on stress in SPECS comes predominantly from diameter measurement errors, a clear path forward presents itself. Correcting the previously identified shortcomings in the system's calibration sheet has the potential to dramatically reduce diametral error, which in turn will decrease stress error.

Chapter 5: Aluminum Experiments

5.1 Methods

A series of cold upsetting tests were conducted on Aluminum 2024-T351, using a similar range of aspect ratios as used in [16]. A sample diameter of 25 mm was chosen to match the smallest of the calibration blocks. A total of 18 samples were machined from 1" Aluminum 2024-T351 round bar in three aspect ratios: $D_o/H_o = 0.5$, 0.8 and 1.0. Samples were measured and weighed before testing, the results of which can be seen in Table 5.1 below. The chemical composition of Aluminum 2024-T351 can be seen in Table 5.2

Sample	Aspect Ratio	H_o [mm]	D_o [mm]	Mass [g]	Bottom/Top Platens
1	1.0	24.92	25.41	35.13	Stainless/4140 Steel
2	1.0	24.88	25.40	35.11	Stainless/4140 Steel
3	1.0	24.93	25.38	35.16	Stainless/4140 Steel
4	1.0	24.96	25.41	35.20	Stainless/4140 Steel
5	1.0	25.00	25.39	35.18	Stainless/4140 Steel
6	1.0	24.98	25.40	35.20	Stainless/4140 Steel
7	0.8	31.15	25.39	43.86	4140 Steel/4140 Steel
8	0.8	31.32	25.48	44.08	4140 Steel/4140 Steel
9	0.8	31.24	25.40	44.02	4140 Steel/4140 Steel
10	0.8	31.30	25.40	44.05	4140 Steel/4140 Steel
11	0.8	31.37	25.40	44.09	4140 Steel/4140 Steel
12	0.8	31.35	25.40	44.08	4140 Steel/4140 Steel
13	0.5	50.06	25.39	70.48	Stainless/4140 Steel
14	0.5	50.01	25.40	70.52	4140 Steel/4140 Steel
15	0.5	49.86	25.38	70.24	4140 Steel/4140 Steel
16	0.5	49.81	25.41	70.32	4140 Steel/4140 Steel
17	0.5	49.83	25.41	70.31	4140 Steel/4140 Steel
18	0.5	50.06	25.40	70.59	4140 Steel/4140 Steel

Table 5.1: Al2024-T351 cylindrical test samples

Al	Cu	Mg	Mn	Fe	Cr	Other Elements
90.7 - 94.7%	3.8 - 4.9%	1.2 - 1.8%	0.3 - 0.9 %	$\leq$ 0.5%	$\leq$ 0.1%	$\leq$ 0.15%

Table 5.2: Al2024-T351 Chemical Composition [15]

The Instron was programmed to load the test samples at a rate of 1.25 kN/s until one of two end conditions were encountered: 50% engineering strain, or sample breakage (detected as a drop in load by 10% of the current peak load).

5.2 Results

5.2.1 Sample Fracture

Without exception, all of the tested samples failed in compression before reaching 50% engineering strain. Figure 5.1 shows Sample 12 after compressive failure. The fracture at approximately a 45° angle matches the results seen in experimental work [20], and finite element analysis [34] throughout the literature.

Figure 5.1: Sample 12, fractured after compaction

Following fracture, samples 13 and 14 broke cleanly into two separate halves, while all other samples remained in one piece. Figures 5.2 and 5.3 provide some clues as to why. As seen in Figure 5.2, the top edge of the bottom half of the factured samples appear to be folded over, creating a flat, smooth surface. This feature was mirrored in the bottom edge of the top half of each fractured sample. This suggested that at the moment of fracture, the sample's fractured halves slide relative to each other along the fracture plane, folding over where the narrow edges come into contact with the compression tooling. To test this theory, vertical lines were drawn on sample 18 prior to compression. As expected Figure 5.3 shows that the lines were broken and shifted along the plane of fracture.

Examination of the fractured samples showed that aside from the top and bottom

(a) Sample 13 (b) Sample 16

Figure 5.2: Fractured samples showing folded material at top edge

Figure 5.3: Sample 18, with vertical lines showing material movement after fracture

edges where material folded over, the fracture plane was extremely smooth, which suggested that the fused samples were most likely held together only by the folded material on the top and bottom of the sample. To test this, the folded material along the bottom of sample 15 was filed off until a clear break between the two halves was seen. After removing this material, the two halves were then easily pried apart, revealing a smooth, glassy fracture plane just like sample 13 and 14. Figure

5.4 shows the fracture surfaces from samples 14 and 15 for comparison.

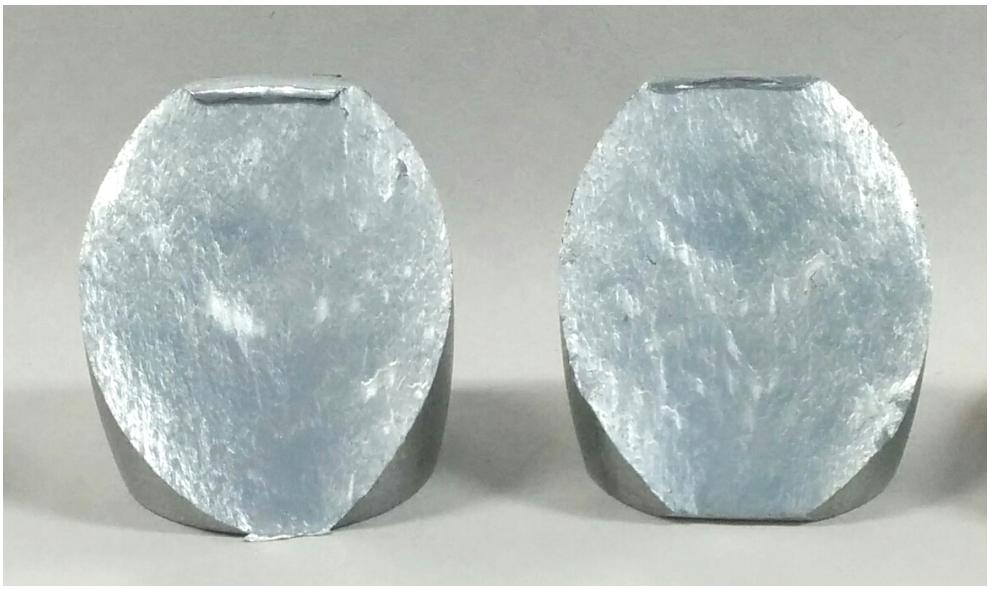


Figure 5.4: Sample 14 (left) and 15 (right) fracture surfaces

5.2.2 Stress-Strain Analysis

True stress-strain curves were developed for all samples, with the exception of sample 1 and sample 4. Sample 1 was excluded from analysis due to work hardening resulting from repeated cold upsetting cycles while first programming the Instron's compression cycle. Sample 4 was excluded from analysis due to a missing tooling ball which prevented proper calculation of sample height. Figures 5.5, 5.7 and 5.9 show the raw force-displacement data collected at each aspect ratio, while Figures 5.6, 5.8 and 5.10 show the stress strain curves developed from this data.

As true stress depends on the instantaneous diameter of the sample, the diameter used in the calculations could have a significant impact on the final result. To illustrate this impact, stress was calculated 3 times for each data point, using the maximum, minimum, and mean measured diameter. True stress-strain curves for Al2024-T351 taken from [20] and [22] are also included for comparison. The values reported in these reports are *not* specific to each aspect ratio, but rather represent the works' overall results.

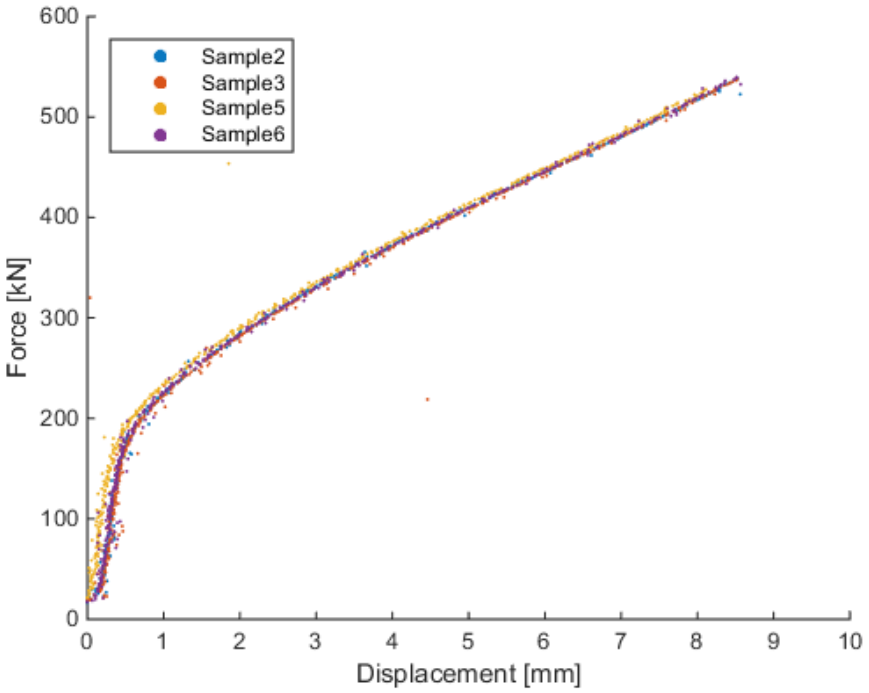


Figure 5.5: Force-Displacement curve for $D_o/H_o = 1.0$

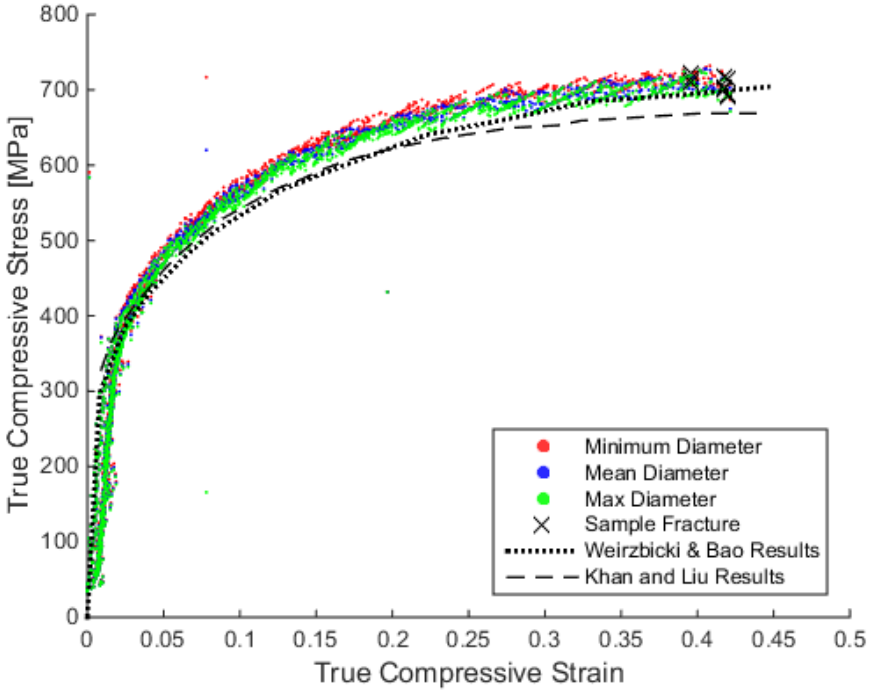


Figure 5.6: True stress-strain curve for $D_o/H_o = 1.0$

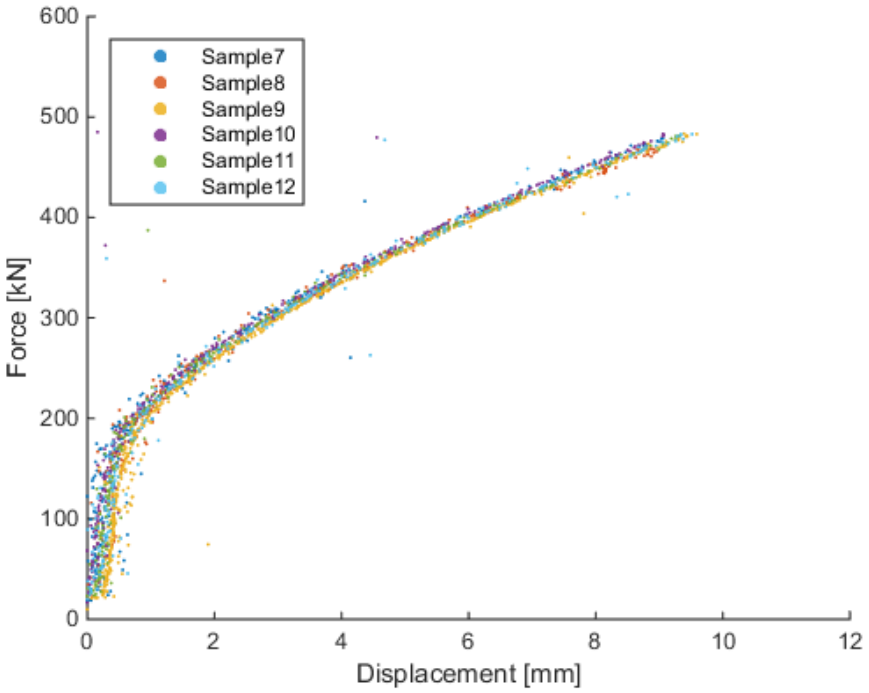


Figure 5.7: Force-Displacement curve for $D_o/H_o = 0.8$

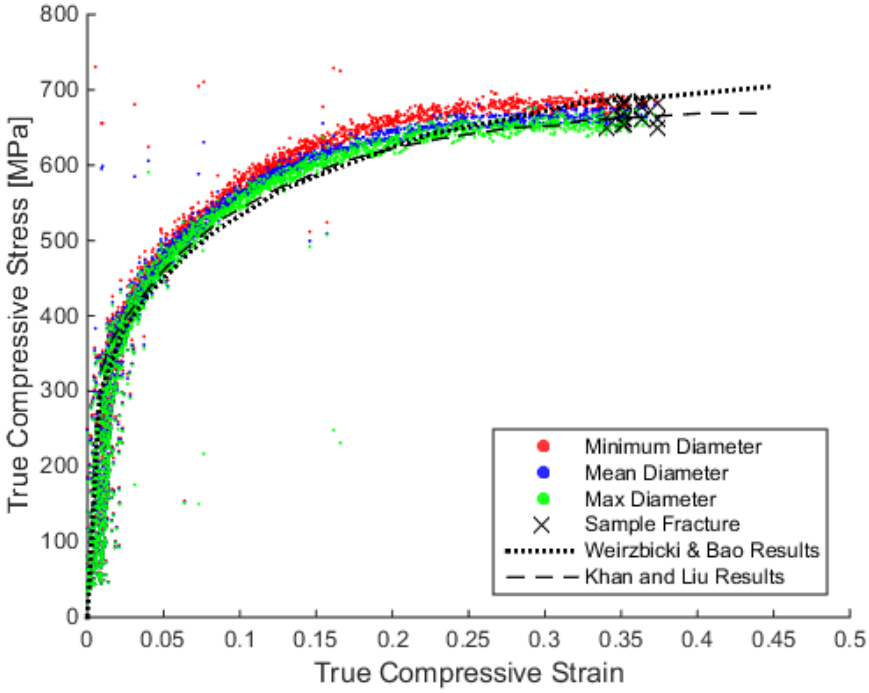


Figure 5.8: True stress-strain curve for $D_o/H_o = 0.8$

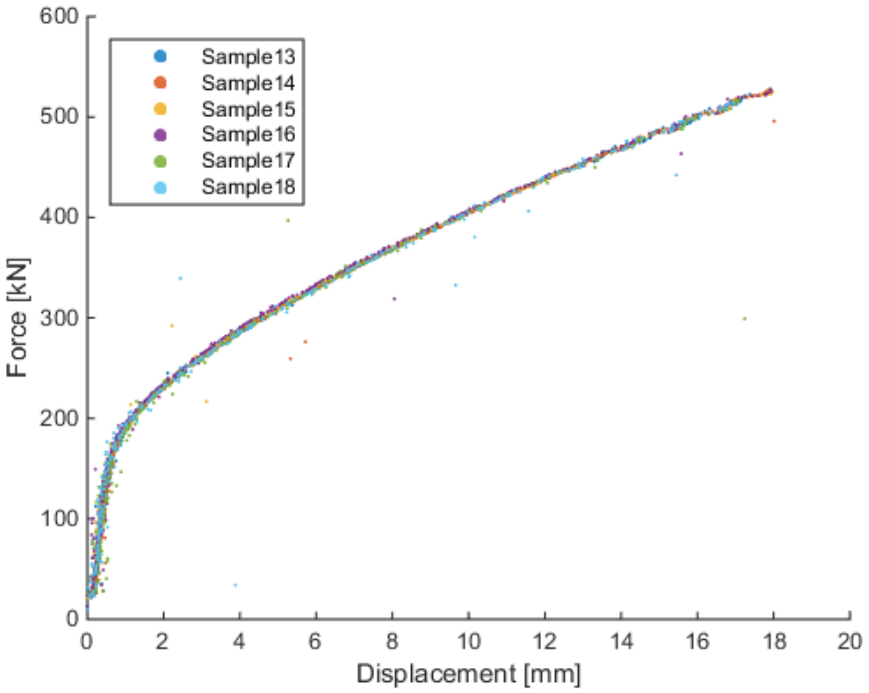


Figure 5.9: Force-Displacement curve for $D_o/H_o = 0.5$

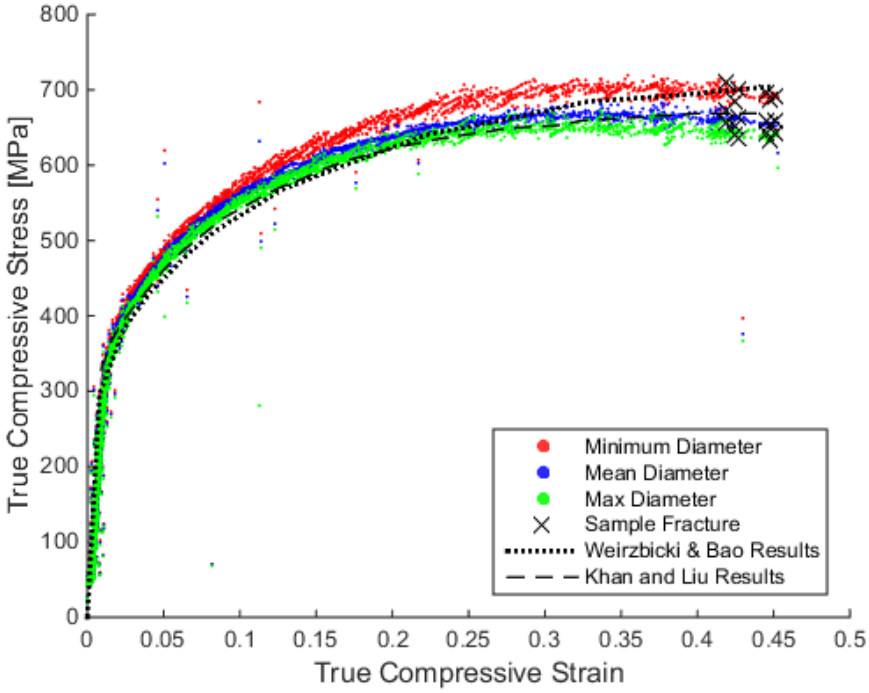


Figure 5.10: True stress-strain curve for $D_o/H_o = 0.5$

The experimental results measured by SPECS agree very well with the literature reported results. Results calculated using minimum, mean, and maximum diameter all follow the general shape of results from literature, and the agreement between literature and experimental results improve with a decreasing aspect ratio (thinner, taller samples).

As would be expected, the results calculated for minimum, mean and maximum diameter agree well with each other to the point where they are almost perfectly overlapped in the elastic deformation region. It is only in the plastic region where barreling effects become significant that the data sets diverge from each other. At this point, a drop in true stress also begins to occur. This drop suggests a build-up of damage within the sample prior to total sample failure.

Figure 5.11 illustrates the effect of SPECS measurement error on mean diameter stress-strain results for one test sample. A smoothed dataset was created for sample 14, taking every tenth datapoint from the raw data. Upper and lower bounds were generated for the error envelope, showing the likely possible range of actual stress-strain values given the measured values. The lower bound was generated by adding e_ϵ and subtracting e_σ (as derived in Section 4.5) from the smoothed data points, shifting the curve down and to the right. The upper bound was created by subtracting e_ϵ and adding e_σ to the smoothed data points, shifting the curve up and to the left.

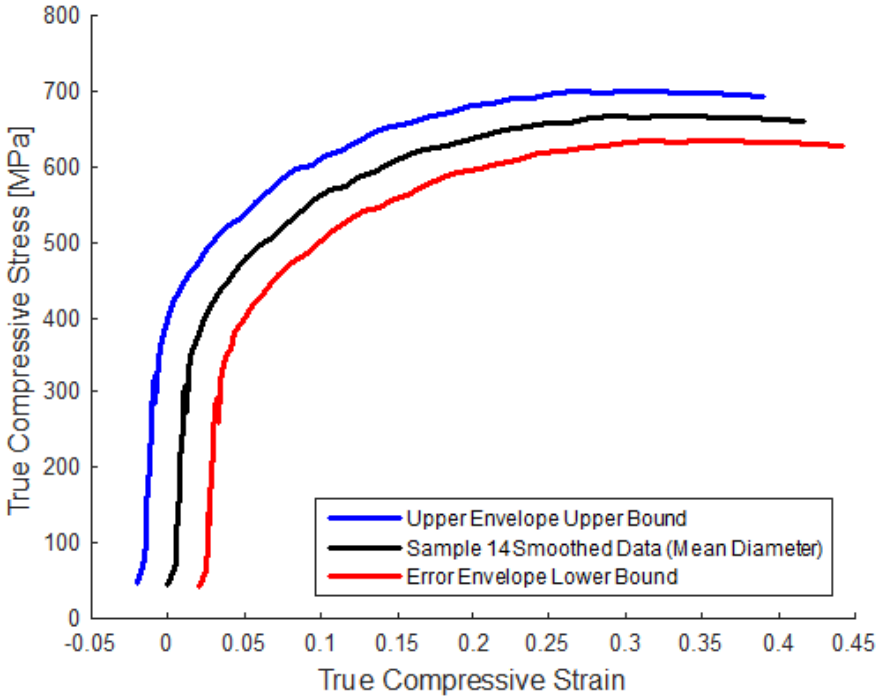


Figure 5.11: Error envelope for Sample 17

Chapter 6: Finite Element Modeling

To further verify the results from testing on Aluminum 2024-T351, a finite element model was developed to simulate the experiments conducted in Chapter 5. The model was developed with LS-DYNA dynamic finite element code, using parameters extracted from SPECS analysis of the experiments.

6.1 Model Overview

The finite element model was developed using the original dimensions of sample 17, one of the tall samples with a nominal aspect ratio (D_o/H_o) of 0.5. The exact dimensions can be found in Table 6.1.

Sample	Aspect Ratio	H_o [mm]	D_o [mm]	Mass [g]	Bottom/Top Platens
17	0.5	49.83	25.413	70.31	4140 Steel/4140 Steel

Table 6.1: Sample 17

The finite element model is built using 2D axisymmetric elements (shell elements using `*SECTION_SHELL`, `ELFORM=15`), and is run using an implicit solution method. The model consists of the three parts seen in Figure 6.1. The upper platen and lower platen (yellow and green elements, respectively) are modeled using the `*MAT_RIGID` material model, which allows for translation but not deformation. The sample (grey elements) is modeled using a `*MAT_POWER_LAW_PLASTICITY` material model.

Contact between the sample and the platens is modeled using two `*CONTACT_AUTOMATIC_SURFACE_TO_SURFACE` cards, one for each platen. Automatic surface-to-surface contact accounts for friction between contact surfaces, and by using two cards, an upper and lower friction coefficient can be set independently. This gives the model more versatility, allowing loading scenarios where the top and bottom platens are made from different materials. There is a 0.1 mm gap between the platens and the sample, to ensure good contact stability at the beginning of a simulation.

The model is loaded by enforcing a vertical displacement on the bottom platen. The bottom platen is moved upwards until the sample is compressed to 50% of its original height. In this simulation, a damage model is not enforced, and sample failure is not observed. Creating a model that remains stable beyond the theoretical breaking strain ensures that valid, stable model results exist for the entire range of experimentally observed strains.

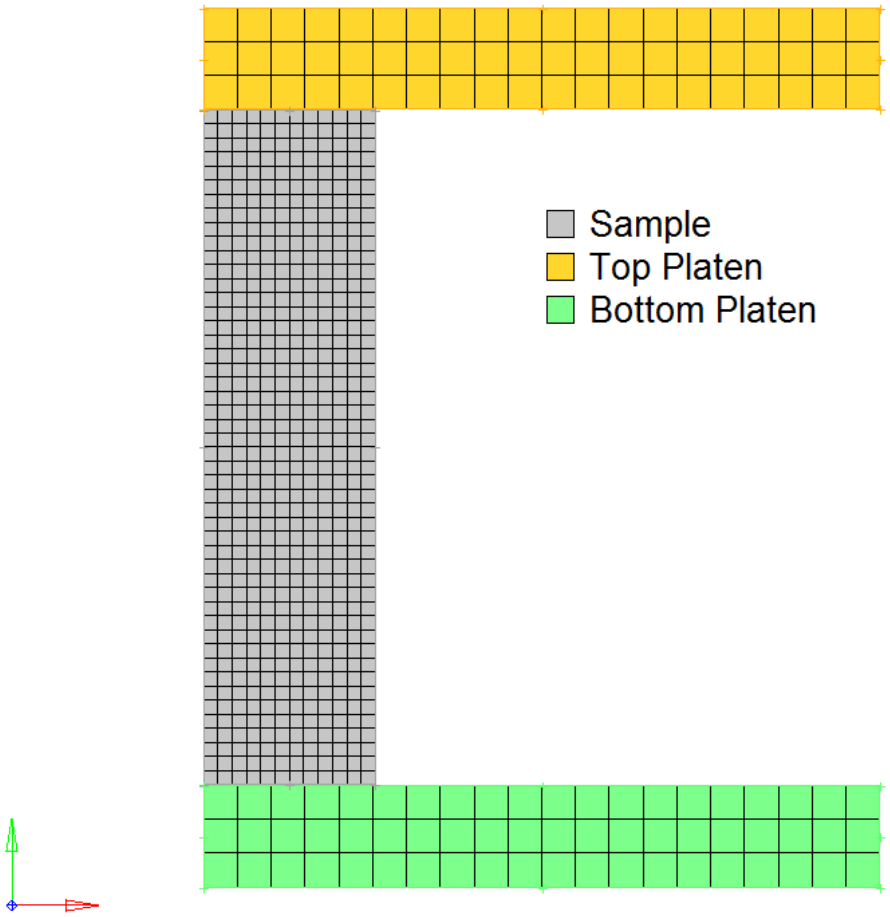


Figure 6.1: Finite element model, 1 mm nominal mesh size

6.2 Material Model

Material parameters for Aluminum 2024-T351 were obtained by fitting the experimental results obtained in Chapter 5 to the piecewise function seen in Equation 6.1. Results from the tallest samples ($D_o/H_o = 0.5$) with stress calculated at the maximum diameter were used for fitting.

$$\sigma_T = \begin{cases} m\epsilon_T + b & \text{for } \epsilon_T \leq \epsilon_{cut}, \\ k\epsilon_T^n & \text{for } \epsilon_T > \epsilon_{cut}. \end{cases} \tag{6.1}$$

where:

σ_T = true stress

ϵ_T = true strain

ϵ_{cut} = cutoff strain between linear and power law segments

m = slope of linear fit

b = offset for linear fit

k = power law strength coefficient

n = power law hardening coefficient

For a given cutoff strain ϵ_{cut}, linear regression was used to fit the coefficients for both parts of the piecewise equation. For the linear segment, the equation can be fit directly using standard linear regression techniques. The power law section first had to be linearized before linear regression could be performed. When linearized, the power law takes the form of Equation 6.2.

$$ln(\sigma_T) = ln(k) + n\, ln(\epsilon_T) \tag{6.2}$$

The fitting procedure was performed for values of ϵ_{cut} ranging from 0 (fully power-law) to 0.4457 (fully linear). R^2 values were calculated as a function of ϵ_{cut} for both halves of the piecewise function in isolation, as well as a combined R^2 for the entire function. The results are shown in Figure 6.2.

All three R^2 values peak around $\epsilon_{cut} = 0.013$. The optimal ϵ_{cut} for each R^2_{cut} and the fitting parameters associated with each are summarized in Table 6.2

R^2 Measure	ϵ_{cut}	m	b	k	n
Power Law Segment	0.01083	26 076	1.398	843.6	0.206
Linear Segment	0.01298	25 669	3.640	828.7	0.196
Combined Fit	0.01290	27 742	3.229	829.4	0.196

Table 6.2: Optimal material fitting parameters for various R^2 measures

When plotted against the original data, the three calculated fits are visually indistinguishable from each other. For the finite element model, the fit calculated for the best combined fit R^2 is used, as it is designed to capture the most accurate

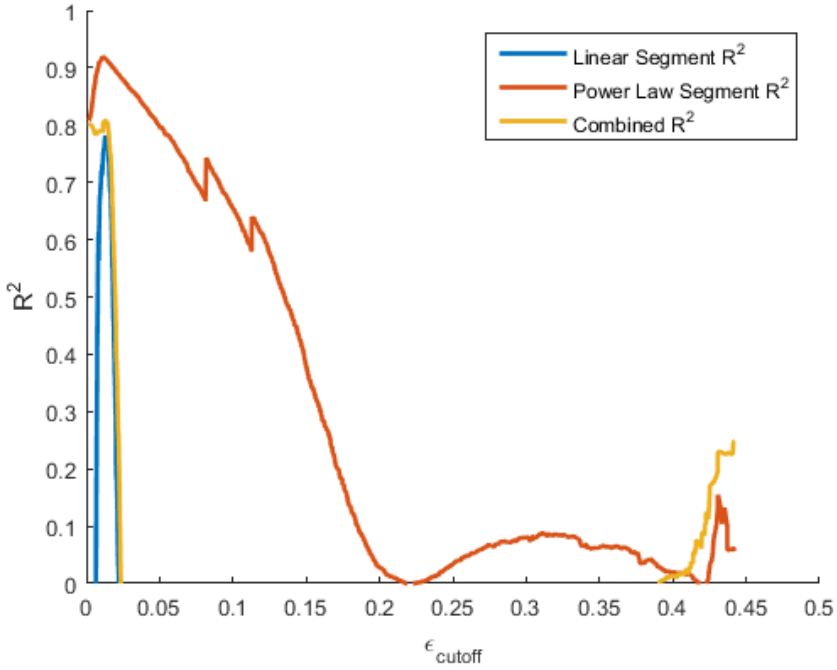


Figure 6.2: Material model fitting: R^2 vs ϵ_{cut}

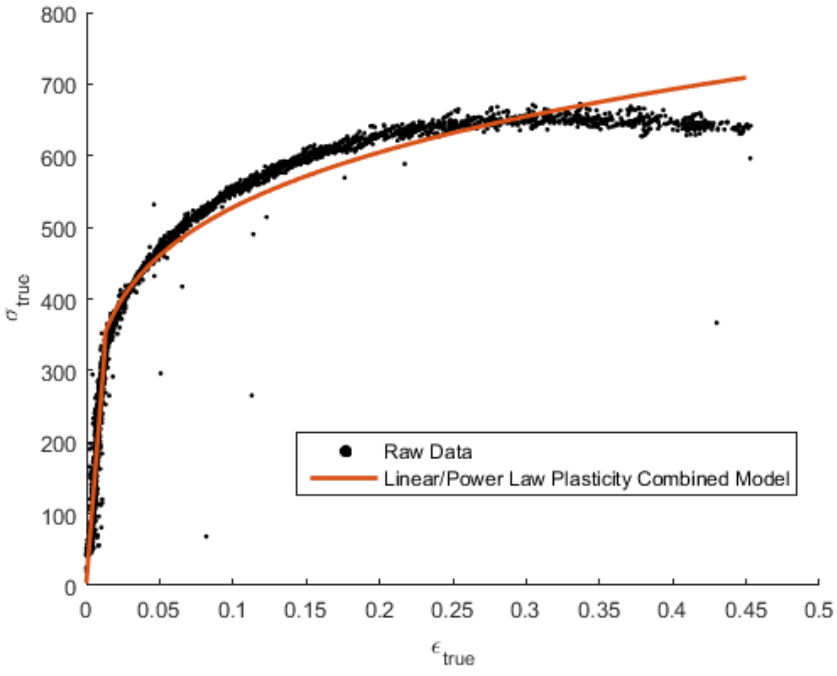


Figure 6.3: Material model fitting: σ_{true} vs ϵ_{true}

material behavior across the widest range of ϵ_T. Figure 6.3 shows the combined linear/power law material model fit plotted against the original fitting data.

The power law fit calculated previously is implemented in the finite element model using LS-DYNA's `*MAT_POWER_LAW_PLASTICITY` material model. In this model, the power law fit is defined using the same k and n parameters calculated previously. The model also requires inputs for Young's modulus (E) and Poisson's ratio (ν), which were obtained from tabulated results on MATWEB for Al2024-T351[15]. The material card has an optional parameter, SIGY, which is used to specify the strain at the yield point of the material. For this simulation, SIGY was set to the cutoff strain $\epsilon_{cut} = 0.0129$. A summary of the input parameters can be found in Table 6.3

k	n	E	ν	SIGY [mm/mm]
829.35 MPa	0.196	73.1 GPa	0.29	0.0129

Table 6.3: Material parameters used in power law plasticity material model

6.3 Friction Coefficient

A friction ring test was used to determine the effective coefficient of friction between Aluminum 2024-T351 and the steel platens used in the experiments in Chapter 5. As mentioned in the literature review in Section 2.3, the equations as-written in [25] fail to reproduce the graphs presented in the original work. The following corrected equations are substituted for calculating μ:

% Decrease in inner diameter ΔD is given by:

$$\Delta D = m \ln\left(\frac{\mu}{0.055}\right) \tag{6.3}$$

where m is given by:

$$\ln m = (0.044 \times \% \text{ height deformation}) + 1.06 \tag{6.4}$$

To the best of the author's knowledge, these corrections have not been published or noted in errata or article notes. However, the original source was available only in print, and a distortion-free image of the calibration curves could not be readily obtained. In the absence of calibration curves that could be accurately interpolated

graphically, the above corrected equations were used as the best approximation available.

For the friction ring tests in this report, Aluminum 2024-T351 rings were machined to be geometrically similar to those used by Male and Cockcroft [25], nominally 24 mm OD × 12 mm ID × 8 mm height. Rings were measured by calliper and weighed before compaction, the results of which can be seen in Table 6.4 below. Four rings were compressed between a pair of 4140 steel platens, and two rings were compressed with a 4140 steel top platen and engraved, hardened stainless steel bottom platen.

The Instron was programmed to load the rings at a rate of 1.25 kN/s until they reached 25% strain. The rings were measured after compaction to determine the % change in inner diameter for the % change in height.

Sample	OD_o [mm]	ID_o [mm]	H_o [mm]	$Mass$ [g]	Bottom Platen	Top Platen
A	24.02	11.90	7.93	7.498	4140 Steel	4140 Steel
B	24.02	11.90	8.00	7.556	4140 Steel	4140 Steel
C	24.03	11.90	7.96	7.536	4140 Steel	4140 Steel
D	24.02	11.89	8.04	7.598	4140 Steel	4140 Steel
E	24.01	11.89	7.96	7.521	Hardened SS	4140 Steel
F	24.03	11.89	7.94	7.515	Hardened SS	4140 Steel

Table 6.4: Aluminum 2024-T351 friction ring test samples

The results of the friction ring tests are shown in Table 6.5, and are plotted alongside the calibration curve in Figure 6.4.

Sample	$H_o[mm]$	$H_f[mm]$	ΔH	$ID_o[mm]$	$ID_f[mm]$	ΔD	m	μ
A*	7.93	6.67	15.89%	11.90	11.35	4.62%	5.81	0.122
B*	8.00	7.56	5.50%	11.90	11.78	1.01%	3.68	0.072
C*	7.96	6.10	23.37%	11.90	10.71	10.00%	8.07	0.190
D*	8.04	6.10	24.13%	11.89	10.67	10.26%	8.35	0.188
E^+	7.96	6.15	22.74%	11.89	11.20	5.80%	7.85	0.115
F^+	7.94	6.12	22.92%	11.89	11.45	3.70%	7.91	0.088

* 4140 steel bottom platen

$^+$ Hardened stainless steel bottom platen

Table 6.5: Fiction ring test results

As seen in Table 6.5, Samples A and B experienced a height change ΔH under 20%, rendering the equations from Male and Cockcroft invalid for these samples [25].

These samples have thus been excluded from analysis and from Figure 6.4. Samples A and B failed to deform the required amount as a result of the sample recovering some of the elastic deformation after compression. Samples C through F were over compressed to account for this springback to result in a final ΔH around 25%.

The results of the friction ring experiment show a small but distinct difference in friction condition between the 4140 steel bottom platen (samples C and D) and the hardened stainless steel platen (samples E and F). All samples show coefficients of friction which are orders of magnitude larger than seen in [2], where results as low as $\mu = 0.04$ were reported for metal samples compressed with teflon film lubrication.

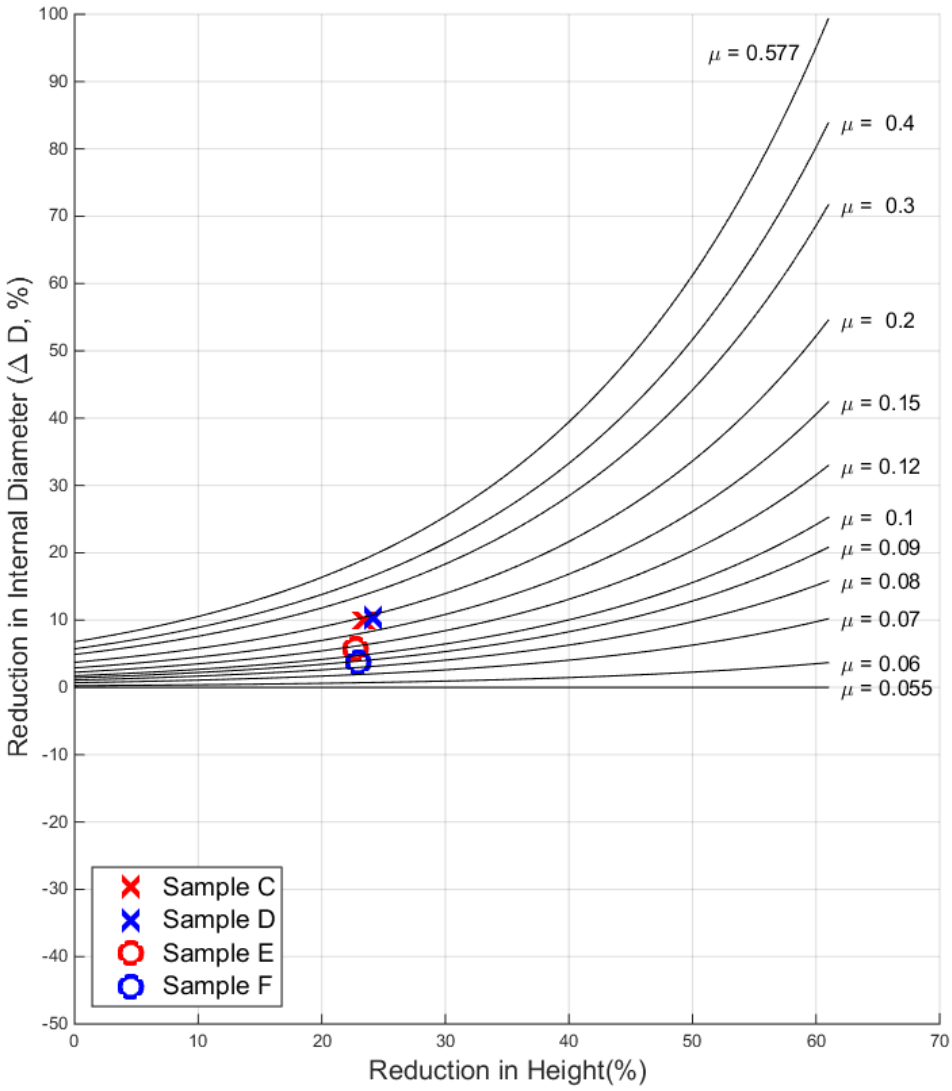


Figure 6.4: Friction ring results plotted on calibration curve

For the finite element model, the contacts were modeled using a coefficient of friction $\mu = 0.189$, the mean of the coefficients found for rings which used 4140 steel platens for both the top and bottom contact surfaces.

6.4 Finite Element Results

A convergence study was conducted to ensure a sufficiently small element size. Starting with a nominal mesh size of 4 mm, the elements of the sample were progressively split, halving in size with each iteration. After running the model, metrics of interest were measured at the final engineering strain of 0.5. Two metrics were chosen for the convergence study: maximum X displacement, and von Mises stress at mid-height along the line of symmetry. The results of the convergence study are shown in Figure 6.5.

Over the course of the convergence study, maximum X displacement did not change significantly, ranging in value from 6.05 to 6.19 mm. Stress at the center of the sample likewise did not change significantly, ranging from 839 to 843 MPa, before converging on a final value of 842 MPa. The final iteration resulted in a nominal mesh size of 0.0625 mm, yielding a total of 147,456 elements. Processing time for this final mesh was approximately three and a half minutes.

FEM results were compared to SPECS observations by plotting the maximum sample diameter versus strain. As seen in Figure 6.6, model results match extremely closely with experimental results, following a similar shape, but with a slight offset.

Stress distribution within the finite element model at the maximum experimental strain ($\epsilon_{eng} = 0.338$) is shown in Figure 6.7. This figure highlights the enormous variation in stress throughout the sample at high strains. From the minimum stress at the center of the sample's faces to the maximum stress at the sample's edges, stress increases nearly threefold.

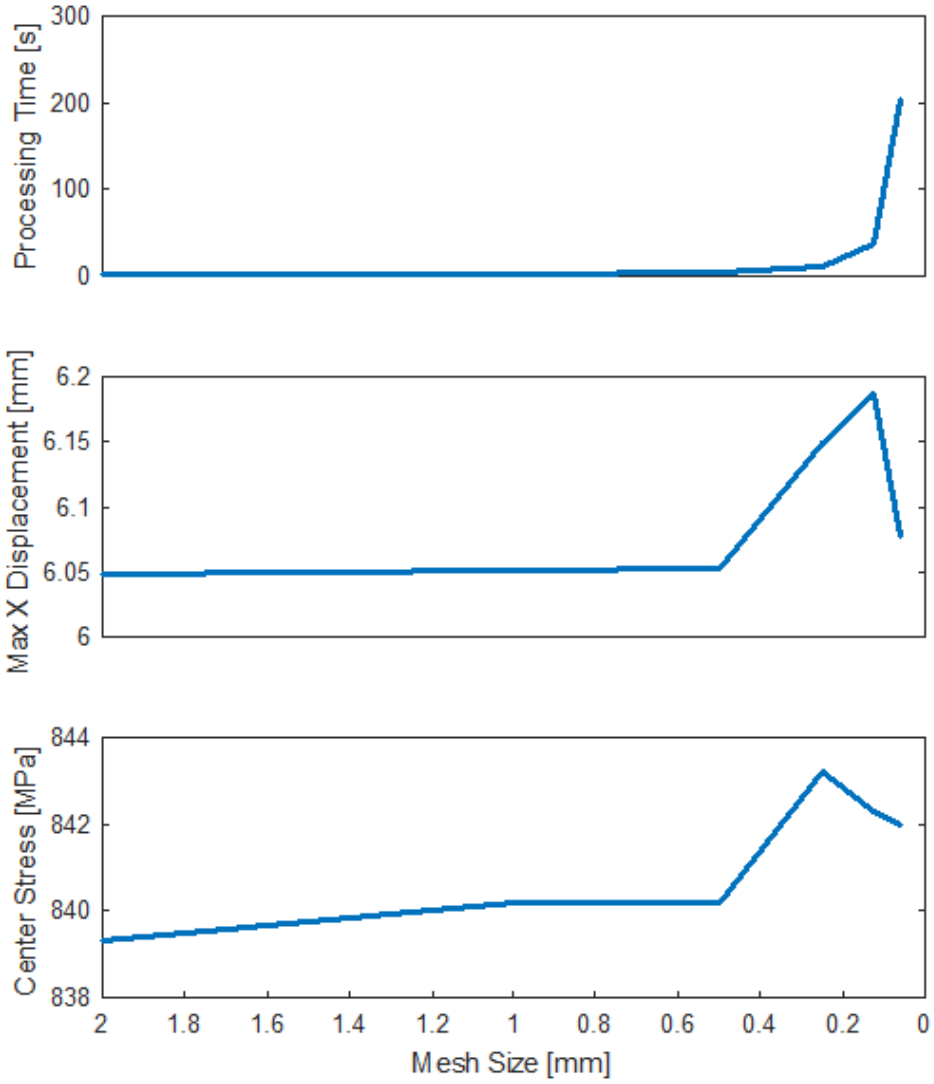


Figure 6.5: Finite element model: convergence study results

Radial displacement, shown in Figure 6.8, follows the trend that would be expected for a barreled sample. Little to no radial displacement is observed towards the center of the sample, and the most extreme radial displacement occurs at the outermost surface. Friction along the sample faces restricts radial displacement compared to the unconstrained material at the sample equator.

A detail view of the top contact surface seen in Figure 6.9 reveals an interesting phenomenon. From the mesh shape, it appears as though material from the side walls

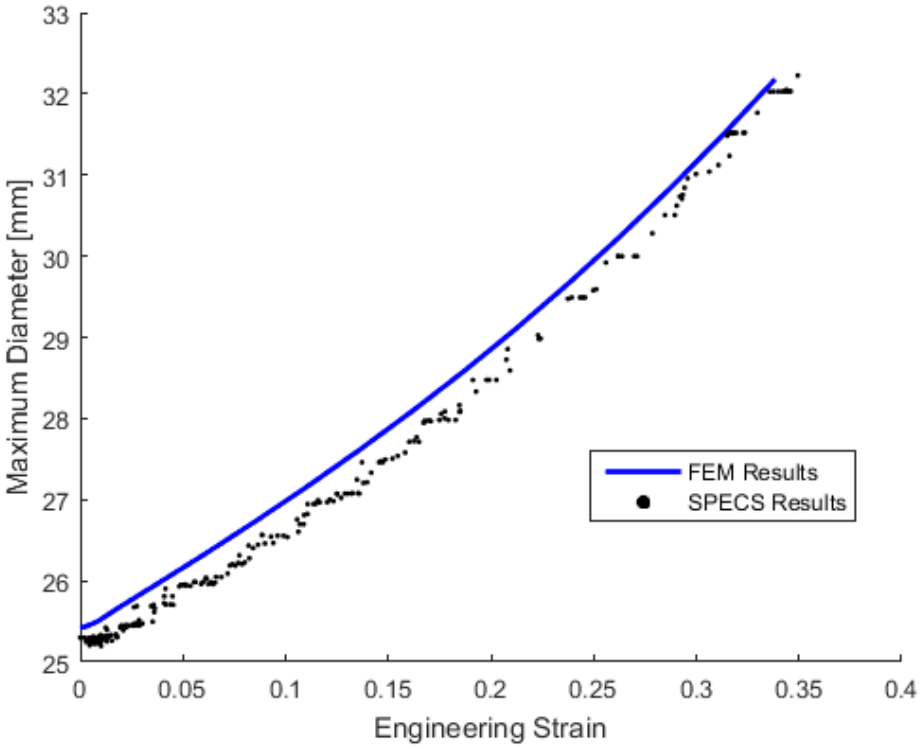


Figure 6.6: SPECS measurements vs FEM results

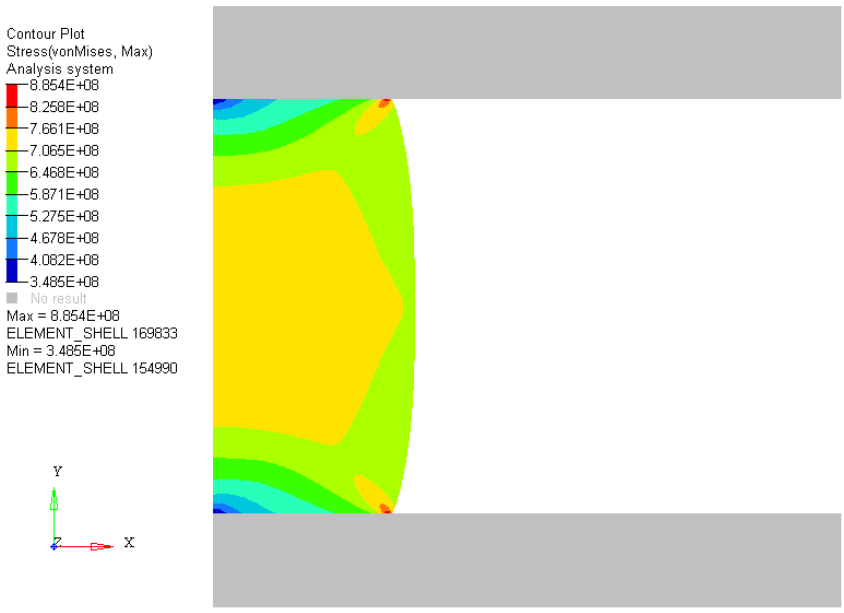


Figure 6.7: Stress distribution in finite element model

has begun to fold over and become incorporated into the top surface of the sample. Taken with the stress distribution in Figure 6.7, this material foldover provides a potential explanation for zones of elevated stress seen near the edge of the sample.

Evidence of this material foldover is seen experimentally, on the top and bottom faces of the test samples after compaction. As shown in Figure 6.10, a distinct ring appears on the top face where foldover is expected to occur.

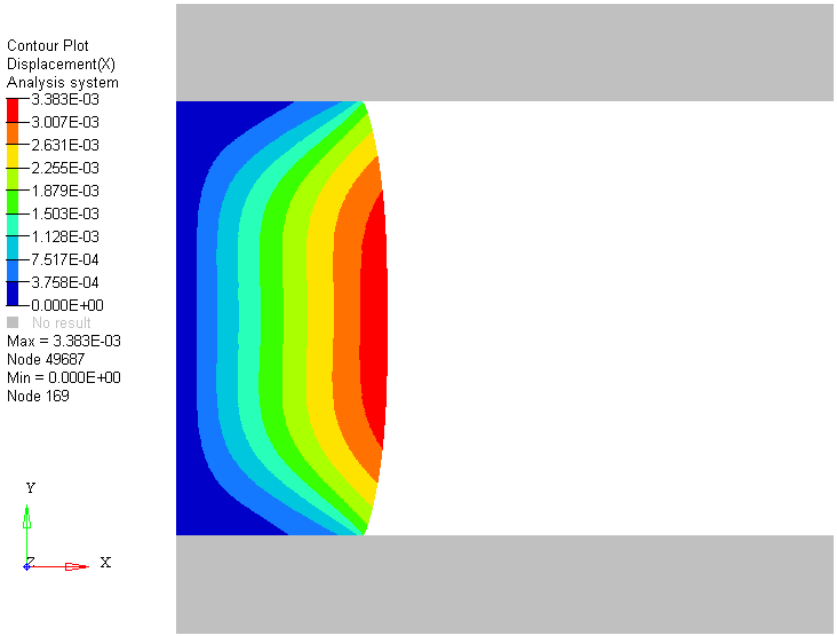


Figure 6.8: Radial displacement in finite element model

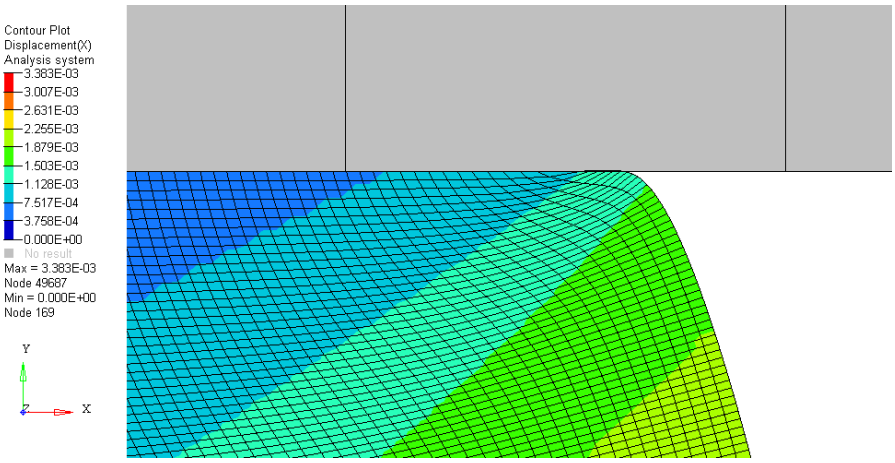


Figure 6.9: Top face of finite element model: detail view

Figure 6.10: Top face of a compacted aluminum sample

Chapter 7: Conclusions and Future Work

7.1 Summary and Conclusions

Video extensometry is a powerful tool for gaining a more complete understanding of how materials respond to load. SPECS, the system detailed in this report, was designed to fill a relatively unexplored niche in video extensometry: a 2D video extensometer which is not restricted to analyzing thin, planar samples. Using the background subtraction techniques described in Chapter 3, the system analyzes the silhouette of a cylindrical test sample to measure its dimensions at mid-plane.

7.1.1 Calibration Experiments

Extensive calibration experiments were conducted to verify the accuracy of the SPECS system. Initial height calibration experiments showed that calibration accuracy is very sensitive to how accurately the spacing between dots on the calibration sheet is measured. It was shown that an accurate calibration dot spacing can be calculated explicitly through regression analysis or measured using microscopy with similar results.

Width calibration experiments revealed a slight bias to SPECS diameter measurements. It was theorized that this bias could be the result of a slight tilt in the calibration sheet. A theoretical analysis showed that a 2° could produce a small but noticeable false magnification of the calibration sheet.

Sources of error were propogated to get an estimate of the error on calculated true stress, true strain values. True strain was found to have a complex error relationship dependent on original and instantaneous sample length. True stress was found to have a relative error of 4.91%.

Overall, the results of the calibration experiments are satisfactory. Stress error exceeds that found in the literature for similar cold upsetting tests by about 2%, with errors associated with diameter measurement accounting for the largest part of this error. These diametral errors should be corrected in future iterations of the system by improving the accuracy and mounting of the calibration sheet.

7.1.2 Aluminum Experiments

A series of cold upsetting tests were conducted using cylinders of Aluminum 2024-T351, using aspect ratios (D_o/H_o) ranging from 0.5 to 1. Qualitatively, observed sample fracture matched very well with predictions from the literature. All samples fractured at roughly 45° angles, leaving a glassy fracture surface on both pieces.

True stress-strain results were plotted with strain calculated using minimum, mean, and maximum diameter. Good agreement was seen between experimental results, and previously established experimental results from the literature.

Overall, the agreement seen between experimental results and those tabulated in the literature is promising. Through these experiments, reasonable stress-strain results were obtained without needing to eliminate the barreling effect.

7.1.3 Finite Element Model

Results from the Aluminum 2024-T351 experiments were used to define a material model for finite element analysis. Using linear regression, experimental results were fitted to a standard power law plasticity material model.

An implicit finite element model was developed to simulate the compaction of sample 17 from the Aluminum 2024-T351 experiments. Convergence of the model was assessed by measuring maximum radial deflection and von Mises stress at the center of the sample. Once converged, results from the finite element model were compared to the original experimental results. Maximum diameter versus engineering strain was very well predicted by the model.

7.2 Future Work

Future work for SPECS can be largely divided into two categories: improvements to the current system and extension of system capabilities.

7.2.1 Improvements to Current System

Efforts to improve the current system should focus on improving the accuracy and resolution of measurements. In particular, significant improvements can be made

by changing the system's calibration procedures. The calibration sheet currently in use has noticeable variance in spacing between dots, which limits the accuracy of the calibration. Furthermore, the calibration as installed currently has a slight tilt, which can be shown to lead to a small but noticeable calibration error. Redesigning the current calibration sheet to have more accurately printed features and be held at a straighter angle has the potential to reduce much of the error seen in the system.

Resolution can also be greatly improved by increasing the size of the test sample in the camera's view. This can be done either through changing lenses on the camera, or by moving the camera closer to the sample. Both options should be explored and evaluated based on cost, ease of implementation, and risk. In particular, care should be taken to ensure that bringing the camera closer to the test sample does not put it at risk of damage from either the Instron press or a fractured sample.

7.2.2 Extension of SPECS Capabilities

The first proposed extension of the current SPECS system is to use the diametral data it outputs to calculate additional parameters related to barreling. In particular, using the diametral data to perform curve fitting should be implemented, so that parameters such as radius of curvature can be calculated as a function of strain.

The second proposed extension of SPECS is to couple the finite element model developed in 6 to the current postprocessing script. The envisioned system would start from estimated material parameters, and refine those parameters on subsequent iterations of the model based on error between model and experimental results. In addition to material parameters, external parameters such as coefficient of friction should also be incorporated into this iterative process.

Finally, now that SPECS has been verified on a well understood anisotropic material (Aluminum 2024-T351), the system can now be used with confidence to analyze more complex materials. In particular, there is great potential to apply SPECS to the measurement of metal foams, powder metallurgy products, and other materials which incorporate large volumes of air. During compaction, the presence of large pockets of air allow for sample volume to change during compaction. A system like SPECS which tracks diameter throughout compaction would be able to track this volume change, leading more accurate input data for material modeling.

Bibliography

[1] A. S. for Testing and M. (ASTM), "Standard test methods of compression testing of metallic materials at room temperature," vol. ASTM Standard E9-09, 2009.

[2] R. Chait and C. H. Curll, "Evaluating engineering alloys in compression," *Recent Developments in Mechanical Testing*, vol. ASTM STP 608, 1976.

[3] Matweb, "Oilite bronze bearing alloy - sae 841," 2016.

[4] F. Yin, W. Tompkins, K. Peterson, and M. Intaglietta, "A video-dimension analyzer," *IEEE Transactions on Biomedical Engineering*, vol. BME-19, 1972.

[5] R. M. Gardner, *Dynamic Aortic Diameter Measurements In Vivo Using Roentgen Videometry*. PhD thesis, University of Utah, 1968.

[6] M. A. Sutton, J. H. Yan, V. Tiwari, H. W. Schreier, and J. J. Orteu, "The effect of out-of-plane motion on 2d and 3d digital image correlation measurements," *Optics and Lasers in Engineering*, vol. 46, 2008.

[7] Instron, "Non-contacting video extensometers," 2016.

[8] F. Schmid, G. Sommer, M. Rappolt, P. Regitnig, G. Holzapfel, P. Laggner, and H. Amenitsch, "Bidirectional tensile testing cell for in situ small angle x-ray scattering investigations of soft tissue," *Nuclear Instruments and Methods in Physics Research B*, vol. 246, 2006.

[9] C. G'Sell, J. Hiver, and A. Dahoun, "Experimental characterization of deformation damage in solid polymers under tension, and its interrelation with necking," *International Journal of Solids and Structures*, vol. 39, 2002.

[10] N. Brusselle-Dupend and L. Cangmi, "A two-phase model for the mechanical behaviour of semicrystalline polymers. part i: Large strains multiaxial validation on hdpe," *Mechanics of Materials*, vol. 40, 2008.

[11] F. Addiego, A. Dahoun, C. G'Sell, and J. Hiver, "Characterization of volume strain at large deformation under uniaxial tension in high-density polyethylene," *Polymer*, vol. 47, 2006.

[12] M. A. Sutton, "Computer vision vased, non-contacting deformation and shape measurements: A revolution in progress," *Journal of the South Carolina Academy of Science*, vol. 11, 2013.

[13] H. Haddadi and S. Belhabib, "Use of rigid-body motion for the investigation and estimation of the measurement errors related to digital imate correlation technique," *Optics and Lasers in Engineering*, vol. 46, 2008.

[14] F. Chen, X. Chen, X. Xie, X. Feng, and L. Yang, "Full-field 3d measurement using multi-camera digital image correlation system," *Optics and Lasers in Engineering*, vol. 51, 2013.

[15] Matweb, "Aluminum 2024-t4, 2024-t351," 2016.

[16] T. Wierzbicki, Y. Bao, Y. Lee, and Y. Bai, "Calibration and evaluation of seven fracture models," *International Journal of Mechanical Sciences*, vol. 47, 2005.

[17] W. Johnson and P. B. Mellor, *Engineering Plasticity*. Ellis Horwood, 1983.

[18] F. Chen and C. Chen, "On the nonuniform deformation of the cylinder compression test," *Transactions of the ASME*, vol. 122, 2000.

[19] J. K. Banerjee, "Barreling of solid cylinders under axial compression," *Transactions of the ASME*, vol. 107, 1985.

[20] Y. Bao and T. Wierzbicki, "A comparative study on various ductile crack formation criteria," *Transactions of the ASME*, vol. 126, 2004.

[21] Y. Bao, *Prediction of Ductile Crack Formation in Uncracked Bodies*. PhD thesis, MIT, 2003.

[22] A. S. Khan and H. Liu, "A new approach for ductile fracture prediction on al2024-t351 alloy," *International Journal of Plasticity*, vol. 35, 2012.

[23] J. Seidt and A. Gilat, "Plastic deformation of 2024-t351 aluminum plate over a wide range of loading conditions," *International Journal of Solids and Structures*, vol. 50, 2013.

[24] M. Kunogi, "A new method of cold extrusion," *Journal of the Scientific Research Institute*, vol. 50, 1956.

[25] A. T. Male and M. Cockcroft, "Coefficient of friction under conditions of bulk plastic deformation," *J. Inst. Metal*, vol. 93, 1964.

[26] H. Sofuoglu and J. Rasty, "On the measurement of friction coefficient utilizing the ring compression test," *Tribology Inernational*, vol. 32, 1999.

[27] C. Hu, H. Ou, and Z. Zhao, "An alternative evaluation method for friction condition in cold forging by ring with boss compression test," *Journal of Materials Processing Technology*, vol. 224, 2-15.

[28] Point Grey Research, *Flea3 USB 3.0 Technical Reference*, 7.0 ed., 9 2014.

[29] N. Instruments, "What are the requirements for an ni vision calibration grid?," 2009.

[30] N. Instruments, "When to use spatial calibration," 2011.

[31] Mathworks, "Least-squares fitting," 2016.

[32] E. Optics, "Understanding focal length and field of view," 2016.

[33] Instron, *Industrial Series RD Models*, 4.0713 ed., 7 2013.

[34] T. S. Cao, "Numerical simulation of 3d ductile cracks formation using recent improved lode-dependent plasticity and damage models combined with remeshing," *International Journal of Solids and Structures*, vol. 51, 2014.

Appendix A: Finite Element Code

Main File

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$        LS-DYNA INPUT DECK                                                    $
$        -----------------------------------------------------------------    $
$        Created by:  Darrel A. Doman, PhD, PEng (Dalhousie University)       $
$        Created on:  Apr-20-2015                                              $
$        Updated by:  Andrea J. Felling, BEng, EIT (Dalhousie University)     $
$        Updated on:  Apr-30-2016                                              $
$        -----------------------------------------------------------------    $
$        DALHOUSIE CONFIDENTIAL & PROPRIETARY                                  $
$        -----------------------------------------------------------------    $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Units
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$   LENGTH      MASS      TIME     FORCE    STRESS    ENERGY     POWER   DENSITY
$      [m]      [kg]       [s]       [N]      [Pa]       [J]       [W]   [kg/m3]
$
$
*KEYWORD 1000M NCPU=8
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                              $
$                            PARAMETER DEFINITIONS                             $
$                                                                              $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*PARAMETER
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$    PRMR1      VAL1     PRMR2      VAL2     PRMR3      VAL3     PRMR4      VAL4
R ENDTIM      +20.00R HLDTIM      +0.030R DTOUT      +0.1000R MAXDISP     0.0252
$    PRMR5      VAL5     PRMR6      VAL6     PRMR7      VAL7     PRMR8      VAL8
```

```
R LOAD          80E6R FRIC       +0.1890R SFACT         1.00
$
$
*TITLE
COLD UPSETTING (1mm Mesh, 0.5AR, 50% strain, .189 mu)
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                             $
$                                 CONTROL CARD                                $
$                                                                             $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*CONTROL_ACCURACY
$---+----1----+----2----+----3
$      OSU       INN    PISOSU
         0         1
$
$
*CONTROL_TERMINATION
$---+----1----+----2----+----3----+----4----+----5
$   ENDTIM    ENDCYC     DTMIN    ENDENG    ENDMAS
&ENDTIM             0       0.0       0.0       0.0
$
$
*CONTROL_ENERGY
$---+----1----+----2----+----3----+----4
$     HGEN      RWEN    SLNTEN     RYLEN
         2         1         2         1
$
$
$*CONTROL_HOURGLASS
$---+----1----+----2
$      IHQ        QH
$         1      0.10
```

```
$
$
*CONTROL_IMPLICIT_GENERAL
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$   IMFLAG       DTO    IMFORM      NSBS       IGS     CNSTN      FORM    ZERO_V
         1     0.100
$
$
*CONTROL_IMPLICIT_SOLUTION
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$  NLSOLVR    ILIMIT    MAXREF     DCTOL     ECTOL     RCTOL     LSTOL    ABSTOL
         2         1
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7
$    DNORM   DIVFLAG   INISTIF   NLPRINT    NLNORM   D3ITCTL     CPCHK

$
$
*CONTROL_IMPLICIT_SOLVER
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$   LSOLVR    LPRINT     NEGEV     ORDER      DRCM    DRCPRM   AUTOSPC   AUTOTOL
         4
$---+----1----+----2
$   LCPACK    MTXDMP
$         2
$
$
*CONTROL_IMPLICIT_AUTO
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$    IAUTO    ITEOPT    ITEWIN     DTMIN     DTMAX     DTEXP     KFAIL    KCYCLE
         1
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                             $
```

```
$                        DATABASE CONTROL FOR BINARY                          $
$                                                                             $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*DATABASE_BINARY_D3PLOT
$---+----1----+----2----+----3----+----4
$  DT/CYCL      LCDT      BEAM     NPLTC
&DTOUT             0         0         0
$---+----1
$    IOOPT
         0
$
$
*DATABASE_GLSTAT
$---+----1----+----2----+----3----+----4----+----5----+----6
$       DT    BINARY      LCUR     IOOPT     DTHFF     BINHF
&DTOUT
$
$
*DATABASE_MATSUM
$---+----1----+----2----+----3----+----4----+----5----+----6
$       DT    BINARY      LCUR     IOOPT     DTHFF     BINHF
&DTOUT
$
$
*DATABASE_RCFORC
$---+----1----+----2----+----3----+----4----+----5----+----6
$       DT    BINARY      LCUR     IOOPT     DTHFF     BINHF
&DTOUT
$
*DATABASE_NODOUT
$---+----1----+----2----+----3----+----4----+----5----+----6
$       DT    BINARY      LCUR     IOOPT     DTHFF     BINHF
&DTOUT                                       0.001
$
```

```
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                             $
$                                  PART CARDS                                 $
$                                                                             $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*INCLUDE_PATH_RELATIVE
../../02 LSDYNA Deck
$
$
*INCLUDE
matlib.k
$
$
*INCLUDE
0_0625mm.mesh
$
$
*INCLUDE
top_platen.part
$
$
*INCLUDE
bottom_platen.part
$
$
*INCLUDE
sample.part
$
$
*SET_PART_LIST
$ TOP PLATEN
$---+----1----+----2----+----3----+----4----+----5----+----6
$      SID       DA1       DA2       DA3       DA4    SOLVER
```

```
        12
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$     PID1      PID2      PID3      PID4      PID5      PID6      PID7      PID8
        2
$
$
*SET_PART_LIST
$ BOTTOM PLATEN
$---+----1----+----2----+----3----+----4----+----5----+----6
$      SID       DA1       DA2       DA3       DA4    SOLVER
        13
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$     PID1      PID2      PID3      PID4      PID5      PID6      PID7      PID8
        3
$
$
*SET_PART_LIST
$ SAMPLE
$---+----1----+----2----+----3----+----4----+----5----+----6
$      SID       DA1       DA2       DA3       DA4    SOLVER
        11
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$     PID1      PID2      PID3      PID4      PID5      PID6      PID7      PID8
        1
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                           $
$                              CONTACT CARDS                                $
$                                                                           $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*CONTACT_2D_AUTOMATIC_SURFACE_TO_SURFACE_TITLE
$---+----1---------------------------------+---------------------------------2
$       ID                                                          HEADING
```

```
      2000                                                  TOP PLATEN-SAMPLE
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7
$     SIDS      SIDM     SFACT      FREQ        FS        FD        DC
        11        12&SFACT              &FRIC
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$   TBIRTH    TDEATH       SOS       SOM       NDS       NDM       COF      INIT

$
$
*CONTACT_2D_AUTOMATIC_SURFACE_TO_SURFACE_TITLE
$---+----1---------------------------------+---------------------------------2
$       ID                                                              HEADING
      1000                                                BOTTOM PLATEN-SAMPLE
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7
$     SIDS      SIDM     SFACT      FREQ        FS        FD        DC
        11        13&SFACT              &FRIC
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$   TBIRTH    TDEATH       SOS       SOM       NDS       NDM       COF      INIT

$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                           $
$                              BOUNDARY SPC CARDS                           $
$                                                                           $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                           $
$                               LOAD CURVE CARDS                            $
$                                                                           $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*DEFINE_CURVE
```

```
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7
$     LCID      SIDR       SFA       SFO      OFFA      OFFO    DATTYP
         1         0      +1.0      +1.0       0.0       0.0         0
$---+----1----+----2----+----3----+----4
$            XVALUES             YVALUES
          +0.000E+00          +0.000E+00
&ENDTIM                         &MAXDISP
           +5000E+00            &MAXDISP
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                            $
$                          LOAD APPLICATION CARDS                            $
$                                                                            $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$*LOAD_SEGMENT_SET_ID
$---+----1---------------------------------+---------------------------------2
$       ID                                                           HEADING
$       100                                                PRESSURE APPLICATION
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$     SSID      LCID        SF        AT        DT
$       100         1
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                            $
$                       DYNAMICS APPLICATION CARDS                           $
$                                                                            $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*BOUNDARY_PRESCRIBED_MOTION_RIGID_ID
$---+----1---------------------------------+---------------------------------2
$       ID                                                           HEADING
       100                                                 PRESSURE APPLICATION
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
```

```
$       PID        DOF        VAD       LCID         SF        VID      DEATH      BIRTH
          3          2          2          1
$
$
*END
```

sample.part

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$        LS-DYNA INPUT DECK                                                     $
$        -----------------------------------------------------------------      $
$        Created by:  Darrel A. Doman, PhD, PEng (Dalhousie University)         $
$        Created on:  Apr-20-2015                                               $
$        Updated by:  Andrea J. Felling, BEng, EIT(Dalhousie University)        $
$        Updated on:  Apr-30-2016                                               $
$        -----------------------------------------------------------------      $
$        DALHOUSIE CONFIDENTIAL & PROPRIETARY                                   $
$        -----------------------------------------------------------------      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*KEYWORD
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                              $
$                                PART KEYWORDS                                 $
$                                                                              $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*PART
$---+----1
$  HEADING
SAMPLE
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$      PID     SECID       MID     EOSID      HGID      GRAV    ADPOPT      TMID
         1        11     ALM04         0         0         0         0         0
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                              $
$                              SECTION KEYWORDS                                $
$                                                                              $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
```

```
*SECTION_SHELL
$---+----1
$  HEADING
$ SAMPLE SHELL
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$    SECID    ELFORM      SHRF       NIP     PROPT   QR/IRID     ICOMP     SETYP
        11        15                   4
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$       T1        T2        T3        T4      NLOC     MAREA      IDOF    EDGSET

$
$
*END
```

bottom_platen.part

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$        LS-DYNA INPUT DECK                                                     $
$        ------------------------------------------------------------------     $
$        Created by:  Darrel A. Doman, PhD, PEng (Dalhousie University)         $
$        Created on:  Apr-20-2015                                               $
$        Updated by:  Andrea J. Felling, BEng, EIT(Dalhousie University)        $
$        Updated on:  Apr-30-2016                                               $
$        ------------------------------------------------------------------     $
$        DALHOUSIE CONFIDENTIAL & PROPRIETARY                                   $
$        ------------------------------------------------------------------     $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*KEYWORD
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                              $
$                                PART KEYWORDS                                 $
$                                                                              $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*PART
$---+----1
$  HEADING
BOTTOM PLATEN
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$      PID     SECID       MID     EOSID      HGID      GRAV    ADPOPT      TMID
         3        13     STL07         0         0         0         0         0
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                              $
$                              SECTION KEYWORDS                                $
$                                                                              $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
```

```
*SECTION_SHELL
$---+----1
$  HEADING
$ BOTTOM PLATEN SHELL
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$    SECID    ELFORM      SHRF       NIP     PROPT   QR/IRID     ICOMP     SETYP
        13        15                   4
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$       T1        T2        T3        T4      NLOC     MAREA      IDOF    EDGSET

$
$
*END
```

top_platen.part

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$        LS-DYNA INPUT DECK                                                     $
$        -----------------------------------------------------------------      $
$        Created by:  Darrel A. Doman, PhD, PEng (Dalhousie University)         $
$        Created on:  Apr-20-2015                                               $
$        Updated by:  Andrea J. Felling, BEng, EIT(Dalhousie University)        $
$        Updated on:  Apr-30-2016                                                   $
$        -----------------------------------------------------------------      $
$        DALHOUSIE CONFIDENTIAL & PROPRIETARY                                   $
$        -----------------------------------------------------------------      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*KEYWORD
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                              $
$                                PART KEYWORDS                                 $
$                                                                              $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*PART
$---+----1
$  HEADING
TOP PLATEN
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$      PID     SECID       MID     EOSID      HGID      GRAV    ADPOPT      TMID
         2        12     STL06         0         0         0         0         0
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                              $
$                              SECTION KEYWORDS                                $
$                                                                              $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
```

```
*SECTION_SHELL
$---+----1
$  HEADING
$ TOP PLATEN SHELL
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$    SECID    ELFORM      SHRF       NIP     PROPT   QR/IRID     ICOMP     SETYP
        12        15                   4
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$       T1        T2        T3        T4      NLOC     MAREA      IDOF    EDGSET

$
$
*END
```

matlib.k

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$        LS-DYNA MATERIAL LIBRARY                                              $
$        ------------------------------------------------------------------   $
$        Created by:  Darrel A. Doman, PhD, PEng (Dalhousie University)       $
$        Created on:  Jun-10-2008                                              $
$        Updated by:  Andrea J. Felling, BEng, EIT(Dalhousie University)      $
$        Updated on:  Apr-30-2016                                              $
$        ------------------------------------------------------------------   $
$        DALHOUSIE CONFIDENTIAL & PROPRIETARY                                  $
$        ------------------------------------------------------------------   $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
*KEYWORD
$ UNITS:
$---+----1----+----2----+----3
$   LENGTH      TIME      MASS
$        m         s        kg
$
$
*COMMENT
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                        COMMENT
----------------------------------------------------------------------
|                                                                    |
|        LS-DYNA MATERIAL LIBRARY                                    |
|                                                                    |
|        Use of this material model library is permitted by          |
|        authorized individuals, groups, or entities. Dalhousie nor  |
|        the authors assume any responsibility for the validity,     |
|        accuracy, or applicability of any results obtained from use |
|        of this library.                                            |
|        DALHOUSIE CONFIDENTIAL & PROPRIETARY                        |
|        Darrel A. Doman, PhD, PEng                                  |
|        Andrea J. Felling, BEng, EIT                                |
```

```
|_____________________________________________________________________|
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                       $
$                  STEEL ALLOY MATERIAL MODELS                          $
$                                                                       $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*MAT_RIGID
$ Title:      STEEL, RIGID, FIXED
$ Ref:
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$      MID        RO         E        PR         N    COUPLE         M     ALIAS
     STL06      7800  200.0E09     0.290
$---+----1----+----2----+----3
$      CMO      CON1      CON2
      +1.0       7.0       7.0
$---+----1----+----2----+----3----+----4----+----5----+----6
$       A1        A2        A3        V1        V2        V3


$
$
*MAT_RIGID
$ Title:      STEEL, RIGID, Y-TRANSLATION FREE
$ Ref:
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$      MID        RO         E        PR         N    COUPLE         M     ALIAS
     STL07      7800  200.0E09     0.290
$---+----1----+----2----+----3
$      CMO      CON1      CON2
      +1.0       6.0       7.0
$---+----1----+----2----+----3----+----4----+----5----+----6
$       A1        A2        A3        V1        V2        V3
```

```
$
$
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$                                                                             $
$                  ALUMINUM ALLOY MATERIAL MODELS                             $
$                                                                             $
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
*MAT_POWER_LAW_PLASTICITY
$ Title:      2024-T351 ALUMINUM
$ Ref:        Andrea's Logbook, #00037 page 90
$---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
$      MID        RO         E        PR         K         N       SRC       SRP
     ALM04      2780  73.10E09     0.290   829.4E6     0.196         0         0
$---+----1----+----2
$     SIGY        VP
   0.01290         0
$
*END
```

Appendix B: MATLAB Code

Main File - Data Parsing

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Postprocessing_RawDataParse__V1_0.m
%
% This script batch processes a set of data folders created by SPECS during
% data collection
%
%
% To run properly, root_input_path must contain the data folders to be
% analyzed, already processed by SPECS labview postprocessing code
%
%
% If analyzing stress strain data, Instron data must be supplied in a
% folder named Instron Data, containing CSV files which share the name of
% the samples (without timestamp in name)
%
%
% All functions follow a standard naming format:
% Postprocessing__NAME_VX_Y.m
%
% Author: Andrea Felling
% Last Edit: May 31, 2016
% Edited By: Andrea Felling
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all
clear all
clc

%% Processing Step Flags
Flag_OpenFiles = 1; % set to 1 if you want OpenFiles function to run
                    % regardless of whether a data file already exists

Flag_Instron = 0;   % set to 1 if you want to use instron data for calculating
                    % loads and other data of interest.  Set to 0 if load
                    % data is unimportant

Flag_Diameter = 1;  % set to 1 if you want to calculate diameters


%% Create File Pointers, check that file structure is proper, then execute
root_input_path = '..\..\03-Data\01-ThesisRawData\Calibration Blocks';
filename_input = 'RawData.mat';

root_output_path = '..\..\03-Data\02-ThesisProcessedData\PostThesis';
filename_output = 'ParsedData_WidthCal.mat';

if exist([root_output_path,filename_output])
    fprintf('\nWARNING: OUTPUT FILE ALREADY EXISTS\n\nPLEASE DELETE/MOVE MANUALLY AND RE-RUN TO
        CONTINUE');
elseif not(exist(root_output_path))
    fprintf('\nWARNING: OUTPUT FOLDER DOES NOT EXIST\n\nPLEASE CREATE MANUALLY AND RE-RUN TO
        CONTINUE');
else
    Postprocessing__DataParse_V2_0(root_input_path,filename_input,root_output_path,
        filename_output,Flag_OpenFiles,Flag_Instron,Flag_Diameter)
end
```

Main File - Postprocessing and Plotting

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Postprocessing_DataProcessingAndPlotting_V1_2.m
%
%   This script reads parsed data (output by
%   Postprocessing_RawDataParse_V1_0.m), compiles stress strain data, and
%   produces the relavent plots
%
%   Author: Andrea Felling
%   Last Edit: May 31, 2016
%   Edited By: Andrea Felling
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% Set Up Workspace
close all
clear all
clc

%% User Variables
ThresholdPlatenTop = 13;
ThresholdPlatenBottom = 13;
PlatenThickness = 12.7;
PlatenFilter = 1; % Set to 1 to filter out data points where there are any missing platen markers
PlatenSpacingFilter = 1; % Set to 1 to filter out data points where platens are too close
    together to produce valid data
SaveOutput = 0; % Set to 1 to have the program save all data to an output file at the end

%% Plotting Info
Samples_25mm = {'Sample2','Sample3','Sample5','Sample6'};
Samples_31mm = {'Sample7','Sample8','Sample9','Sample10','Sample11','Sample12'};
Samples_50mm = {'Sample13','Sample14','Sample15','Sample16','Sample17','Sample18'};
Samples_Cal = {'SampleCal1','SampleCal3','SampleCal2','SampleCal4'};
NominalDiameter_Cal = {25.13,50.07,38.18,63.19}; %nominal max diameter of cal samples in mm,
    ordered as above
Samples_HeightCal = {'00','01','02','03','04','05','06','07','08','09',...
                     '10','11','12','13','14','15','16','17','18','19',...
                     '20','22','24','26','28','30','32','34','36','38',...
                     '40','42','44','46','48','50','52','54','56','58'};
%% Open Parsed Data
tic
root_path = '..\..\03-Data\02-ThesisProcessedData\PostThesis';
path_Samples = [root_path,'\ParsedData_Samples.mat'];
path_WidthCal = [root_path,'\ParsedData_WidthCal.mat'];
path_Bao = [root_path,'\LitData_Bao.mat'];
path_Khan = [root_path,'\LitData_Khan.mat'];
path_HeightCal = [root_path,'\ParsedData_HeightCal.mat'];
path_HeightCalExcel =[root_path,'\InstronData_HeightCal.xlsx'];

if exist(path_Samples,'file')
    fprintf('\nProcessedData.mat found...\nLoading sample data...')
    % The following three lines load the data into a temporary variable,
    % then assign the variable a new name so that the input file can
    % contain a variable with any name, but still work.  This line is
    % repeated for all data loading steps.
    loaded_data = load(path_Samples);
    VarName = fieldnames(loaded_data);
    Data_Samples = loaded_data(1).(VarName{1});
    fprintf('\nDone!\n')
else
    fprintf('\nERROR: Sample Data Not Found\n')
    return
end
```

```
if exist(path_WidthCal,'file')
    fprintf('\nProcessedData.mat found...\nLoading width calibration data...')
    loaded_data = load(path_WidthCal);
    VarName = fieldnames(loaded_data);
    Data_WidthCal = loaded_data(1).(VarName{1});
    fprintf('\nDone!\n')
else
    fprintf('\nERROR: Width Calibration Data Not Found\n')
    return
end
if exist(path_HeightCal,'file')
    fprintf('\nProcessedData.mat found...\nLoading height calibration data...')
    loaded_data = load(path_HeightCal);
    VarName = fieldnames(loaded_data);
    Data_HeightCal = loaded_data(1).(VarName{1});
    fprintf('\nDone!\n')
else
    fprintf('\nERROR: Height Calibration Data Not Found\n')
    return
end
if exist(path_HeightCalExcel,'file')
    fprintf('\nHeight Calibration Excel Data found...\nLoading data...')
    Data_HeightCalExcel = xlsread(path_HeightCalExcel);
    fprintf('\nDone!\n')
else
    fprintf('\nERROR: Height Calibration Excel Data Not Found\n')
    return
end
if exist(path_Bao,'file')
    fprintf('\nBaoData.mat found...\nLoading theoretical data...')
    loaded_data = load(path_Bao);
    VarName = fieldnames(loaded_data);
    Data_Bao = loaded_data(1).(VarName{1});
    fprintf('\nDone!\n')
else
    fprintf('\nERROR: Bao Data Not Found\n')
    return
end
if exist(path_Khan,'file')
    fprintf('\nData_Khan.mat found...\nLoading theoretical data...')
    loaded_data = load(path_Khan);
    VarName = fieldnames(loaded_data);
    Data_Khan = loaded_data(1).(VarName{1});
    fprintf('\nDone!\n')
else
    fprintf('\nERROR: Khan Data Not Found\n')
    return
end


Data_all = [Data_Samples,Data_HeightCal,Data_WidthCal];

toc

%% Remove Frames with No Diameter Data
fprintf('\n\nRemoving frames without diameter data...')
tic
for n = 1:size(Data_all,2)
    RemoveIndex = size(Data_all(n).Diameter,1)+1;
    Data_all(n).Frames(RemoveIndex:end,:) = [];
    Data_all(n).Load(RemoveIndex:end,:) = [];
    Data_all(n).Circles(RemoveIndex:end,:) = [];
    Data_all(n).Platens(RemoveIndex:end,:) = [];
    Data_all(n).LeftEdge(RemoveIndex:end,:) = [];
    Data_all(n).RightEdge(RemoveIndex:end,:) = [];
end
```

```
fprintf('\nDone!\n')
toc

%% Reprocess Circle Detections if Flag is On
if PlatenFilter==1;
    fprintf('\n\nRemoving frames with missing platen data...')
    tic
    for n = 1:size(Data_all,2)
        for i = 1:size(Data_all(n).Circles,1)
            if isnan(Data_all(n).Circles{i,'TopLeft_X_mm'}) || isnan(Data_all(n).Circles{i,'
                TopRight_X_mm'})
                Data_all(n).Platens{i,'TopPlaten_mm'}=NaN;
            end
            if isnan(Data_all(n).Circles{i,'BottomLeft_X_mm'}) || isnan(Data_all(n).Circles{i,'
                BottomRight_X_mm'})
                Data_all(n).Platens{i,'TopPlaten_mm'}=NaN;
            end
        end
        if (sum(isnan(Data_all(n).Platens{:,'TopPlaten_mm'}))==size(Data_all(n).Platens,1)) || (
            sum(isnan(Data_all(n).Platens{:,'BottomPlaten_mm'}))==size(Data_all(n).Platens,1))
            invalid_tests(n,1) = 1;
            fprintf('\n    >ERROR: %s has no valid platen data after filtering\n    >Removing
                ...',Data_all(n).SampleName);
        else
            invalid_tests(n,1) = 0;
        end
    end
    for n = size(Data_all,2):-1:1
        if invalid_tests(n,1)==1
            Data_all(n)=[];
        end
    end

    fprintf('\nDone!\n')
    toc
end
%% Reprocess Platen Data if Spacing Flag is On
frames_removed = 0;
if PlatenSpacingFilter==1;
    fprintf('\n\nRemoving frames with platens too close for valid data...')
    tic
    MinPlatenSpacing = ThresholdPlatenTop + ThresholdPlatenBottom;
    for n = 1:size(Data_all,2)
        for i = size(Data_all(n).Platens,1):-1:1
            if abs(Data_all(n).Platens{i,'TopPlaten_mm'} ...
                    - Data_all(n).Platens{i,'BottomPlaten_mm'}) <= MinPlatenSpacing

                Data_all(n).Frames(i,:) = [];
                Data_all(n).Load(i,:) = [];
                Data_all(n).Circles(i,:) = [];
                Data_all(n).Platens(i,:) = [];
                Data_all(n).LeftEdge(i,:) = [];
                Data_all(n).RightEdge(i,:) = [];
                Data_all(n).Diameter(i,:) = [];
                frames_removed = frames_removed+1;
            end
        end
    end

    fprintf('\nDone!\n')
    toc
end

%% Compile Data into Aligned Table
fprintf('\n\nCompiling data into frame-number aligned tables...')
tic
```

```
for n = 1:size(Data_all,2)
    Data_all(n).Compiled = join(Data_all(n).Frames, Data_all(n).Load);
    Data_all(n).Compiled = join(Data_all(n).Compiled, Data_all(n).Platens);
    Data_all(n).Compiled = join(Data_all(n).Compiled, Data_all(n).Diameter);


    Data_all(n).Compiled.Properties.VariableNames{'Load_N'} = 'Load';
    Data_all(n).Compiled.Properties.VariableNames{'TopPlaten_mm'} = 'TopPlaten';
    Data_all(n).Compiled.Properties.VariableNames{'BottomPlaten_mm'} = 'BottomPlaten';
    Data_all(n).Compiled.Properties.VariableNames{'Height_mm'} = 'Y';
    Data_all(n).Compiled.Properties.VariableNames{'Left_mm'} = 'LeftX';
    Data_all(n).Compiled.Properties.VariableNames{'Right_mm'} = 'RightX';
    Data_all(n).Compiled.Properties.VariableNames{'Diameter_mm'} = 'Diameter';
end
fprintf('\nDone!\n')
toc

%% Remove NaN Load and displacement frames from data (data taken outside of Instron test)
fprintf('\n\nDeleting frames without load data...')
tic
for n = 1:size(Data_all,2)
    NaNRows = isnan(Data_all(n).Compiled{:,'Load'});
    Data_all(n).Compiled = Data_all(n).Compiled(not(NaNRows),:);
end
fprintf('\nDone!\n')
toc

fprintf('\n\nDeleting frames without displacement data...')
tic
for n = 1:size(Data_all,2)
    NaNRows = isnan(Data_all(n).Compiled{:,'TopPlaten'});
    Data_all(n).Compiled = Data_all(n).Compiled(not(NaNRows),:);
    NaNRows = isnan(Data_all(n).Compiled{:,'BottomPlaten'});
    Data_all(n).Compiled = Data_all(n).Compiled(not(NaNRows),:);
end
fprintf('\nDone!\n')
toc

%% Isolate sample edges from other edge detections
fprintf('\n\nIsolating sample edges and diameters from each frame...')
tic
for n = 1:size(Data_all,2)
    SampleEdges=table();
    for i = 1:size(Data_all(n).Compiled,1)
        SampleRows = (Data_all(n).Compiled.Y{i,1}>Data_all(n).Compiled{i,'TopPlaten'}+
            ThresholdPlatenTop) & ...
                (Data_all(n).Compiled.Y{i,1}<Data_all(n).Compiled{i,'BottomPlaten'}-
                    ThresholdPlatenBottom);

        SampleEdges(i,:) = table((Data_all(n).Compiled{i,'FrameNumber'}),...
                        {Data_all(n).Compiled.Y{i,1}(SampleRows)},...
                        {Data_all(n).Compiled.LeftX{i,1}(SampleRows)},...
                        {Data_all(n).Compiled.RightX{i,1}(SampleRows)},...
                        {Data_all(n).Compiled.Diameter{i,1}(SampleRows)},...
                          mean(Data_all(n).Compiled.Diameter{i,1}(SampleRows)),...
                          min(Data_all(n).Compiled.Diameter{i,1}(SampleRows)),...
                          max(Data_all(n).Compiled.Diameter{i,1}(SampleRows)));



    end
    SampleEdges.Properties.VariableNames = {'FrameNumber','SampleY','SampleLeft','SampleRight','
        SampleDiameter','MeanDiameter','MinDiameter','MaxDiameter'};
    Data_all(n).Compiled = join(Data_all(n).Compiled,SampleEdges);
end
fprintf('\nDone!\n')
```

```
toc

%% Calculate Stress
fprintf('\n\nCalculating Stress...')
tic
for n = 1:size(Data_all,2)
    SampleStress=table();
    for i = 1:size(Data_all(n).Compiled,1)
        MeanDiamTrueStress(i,:) = Data_all(n).Compiled{i,'Load'}/(0.25*pi*(Data_all(n).Compiled{i
            ,'MeanDiameter'}/1000)^2);
        MinDiamTrueStress(i,:) = Data_all(n).Compiled{i,'Load'}/(0.25*pi*(Data_all(n).Compiled{i
            ,'MinDiameter'}/1000)^2);
        MaxDiamTrueStress(i,:) = Data_all(n).Compiled{i,'Load'}/(0.25*pi*(Data_all(n).Compiled{i
            ,'MaxDiameter'}/1000)^2);
        EngStress(i,:) = Data_all(n).Compiled{i,'Load'}/(0.25*pi*(Data_all(n).Compiled{1,'
            MeanDiameter'}/1000)^2);
        SampleStress(i,:) = table((Data_all(n).Compiled{i,'FrameNumber'}),...
                                  MeanDiamTrueStress(i,1)*10^-6,...
                                  MinDiamTrueStress(i,1)*10^-6,...
                                  MaxDiamTrueStress(i,1)*10^-6,...
                                  EngStress(i,:)*10^-6);
    end
    SampleStress.Properties.VariableNames = {'FrameNumber','MeanDiamTrueStress','
        MinDiamTrueStress','MaxDiamTrueStress','EngStress'};
    Data_all(n).Compiled = join(Data_all(n).Compiled,SampleStress);
    Data_all(n).Compiled.Properties.VariableUnits{'MeanDiamTrueStress'} = 'MPa';
    Data_all(n).Compiled.Properties.VariableUnits{'MinDiamTrueStress'} = 'MPa';
    Data_all(n).Compiled.Properties.VariableUnits{'MaxDiamTrueStress'} = 'MPa';
    Data_all(n).Compiled.Properties.VariableUnits{'EngStress'} = 'MPa';


end
fprintf('\nDone!\n')
toc

%% Calcuate Displacement and Compressive True/Engineering Strain
fprintf('\n\nCalculating Strain...')
tic
for n = 1:size(Data_all,2)
    SampleHeight = Data_all(n).Compiled{:,'BottomPlaten'}-Data_all(n).Compiled{:,'TopPlaten'}-
        PlatenThickness;
    SampleDisplacement = SampleHeight - SampleHeight(1);
    SampleEngStrain = -(SampleDisplacement./SampleHeight(1));
    SampleTrueStrain = -(log(SampleHeight./SampleHeight(1)));

    SampleStrain = table(Data_all(n).Compiled{:,'FrameNumber'},...
                         SampleHeight,...
                         SampleDisplacement,...
                         SampleEngStrain,...
                         SampleTrueStrain);
    SampleStrain.Properties.VariableNames = {'FrameNumber','SampleHeight','SampleDisplacement','
        EngStrain','TrueStrain'};

    Data_all(n).Compiled = join(Data_all(n).Compiled,SampleStrain);
    Data_all(n).Compiled.Properties.VariableUnits{'SampleHeight'} = 'mm';

end
fprintf('\nDone!\n')
toc

%% Calcuate Displacement and Compressive True/Engineering Strain
fprintf('\n\nCalculating Fracture Points...')
tic
for n = 1:size(Data_all,2)
    SampleHeight = Data_all(n).Compiled{:,'BottomPlaten'}-Data_all(n).Compiled{:,'TopPlaten'}-
        PlatenThickness;
```

```
    SampleDisplacement = SampleHeight - SampleHeight(1);
    SampleEngStrain = -(SampleDisplacement./SampleHeight(1));
    SampleTrueStrain = -(log(SampleHeight./SampleHeight(1)));

    SampleStrain = table(Data_all(n).Compiled{:,'FrameNumber'},...
                         SampleHeight,...
                         SampleDisplacement,...
                         SampleEngStrain,...
                         SampleTrueStrain);
    SampleStrain.Properties.VariableNames = {'FrameNumber','SampleHeight','SampleDisplacement','
        EngStrain','TrueStrain'};

    Data_all(n).Compiled = join(Data_all(n).Compiled,SampleStrain);
    Data_all(n).Compiled.Properties.VariableUnits{'SampleHeight'} = 'mm';

end
fprintf('\nDone!\n')
toc


%% Platen Offset Image
fprintf('\n\nCreating Calibration Data Graphics...')
tic
X1 = Data_all(1).LeftEdge.X_mm{383,1};
Y1 = Data_all(1).LeftEdge.Y_mm{383,1};
Y2 = Data_all(1).RightEdge.Y_mm{383,1};
X2 = Data_all(1).RightEdge.X_mm{383,1};
TopLine=Data_all(1).Compiled{291,'TopPlaten'}+ThresholdPlatenTop;
BottomLine=Data_all(1).Compiled{291,'BottomPlaten'}-ThresholdPlatenTop;
figure()
hold on
plot(X1,180-Y1,'b.')
plot(X2,180-Y2,'r.')
plot([25 235],180-ones(1,2)*TopLine,'k:','LineWidth',1.5)
plot([25 235],180-ones(1,2)*BottomLine,'k:','LineWidth',1.5)
axis equal
xlabel('X [mm]')
ylabel('Y [mm]')
legend('Left Edge','Right Edge','Region of Interest','Location','SouthEast')

%% Sample Index List
for i = 1:size(Samples_50mm,2)
    SampleIndex_50mm(i,1) = find(arrayfun(@(n) strcmp(Data_all(n).SampleName, Samples_50mm(i)),
        1:numel(Data_all)));
end
for i = 1:size(Samples_31mm,2)
    SampleIndex_31mm(i,1) = find(arrayfun(@(n) strcmp(Data_all(n).SampleName, Samples_31mm(i)),
        1:numel(Data_all)));
end
for i = 1:size(Samples_25mm,2)
    SampleIndex_25mm(i,1) = find(arrayfun(@(n) strcmp(Data_all(n).SampleName, Samples_25mm(i)),
        1:numel(Data_all)));
end
for i = 1:size(Samples_Cal,2)
    CalIndex(i,1) = find(arrayfun(@(n) strcmp(Data_all(n).SampleName, Samples_Cal(i)), 1:numel(
        Data_all)));
end
for i = 1:size(Samples_HeightCal,2)
    HeightCalIndex(i,1) = find(arrayfun(@(n) strcmp(Data_all(n).SampleName, Samples_HeightCal(i))
        , 1:numel(Data_all)));
end
SampleIndex = [SampleIndex_25mm;SampleIndex_31mm;SampleIndex_50mm];

%% Height Calibration Analysis
clear HeightCalibrationStats
for i = 1:size(HeightCalIndex,1)
```

```
    n = HeightCalIndex(i,1);
    HeightCalibrationStats(i).SampleName = Data_all(n).SampleName;
    HeightCalibrationStats(i).PlatenBottom = Data_all(n).Compiled{1:8,'BottomPlaten'};
    HeightCalibrationStats(i).PlatenTop = Data_all(n).Compiled{1:8,'TopPlaten'};

    excel_row = Data_HeightCalExcel(:,1) == str2num(HeightCalibrationStats(i).SampleName);
    HeightCalibrationStats(i).HeightGaugeRaw = mean(Data_HeightCalExcel(excel_row,3:5));
    HeightCalibrationStats(i).InstronRaw = Data_HeightCalExcel(excel_row,2);
    HeightCalibrationStats(i).HeightGauge = HeightCalibrationStats(i).HeightGaugeRaw -
        HeightCalibrationStats(1).HeightGaugeRaw;
    HeightCalibrationStats(i).Instron = HeightCalibrationStats(i).InstronRaw-
        HeightCalibrationStats(1).InstronRaw;
    HeightCalibrationStats(i).BottomDisplacement = mean(HeightCalibrationStats(1).PlatenBottom) -
         HeightCalibrationStats(i).PlatenBottom;
    HeightCalibrationStats(i).TopDisplacement = mean(HeightCalibrationStats(1).PlatenTop) -
        HeightCalibrationStats(i).PlatenTop;


    HeightCalibrationStats(i).BottomError_abs = HeightCalibrationStats(i).BottomDisplacement -
        HeightCalibrationStats(i).HeightGauge;
    HeightCalibrationStats(i).TopError_abs = HeightCalibrationStats(i).TopDisplacement -
        HeightCalibrationStats(i).HeightGauge;
    HeightCalibrationStats(i).InstronError_abs = HeightCalibrationStats(i).Instron -
        HeightCalibrationStats(i).HeightGauge;

    HeightCalibrationStats(i).TopError_rel = (HeightCalibrationStats(i).TopError_abs ./
        HeightCalibrationStats(i).HeightGauge)*100;
    HeightCalibrationStats(i).BottomError_rel = (HeightCalibrationStats(i).BottomError_abs ./
        HeightCalibrationStats(i).HeightGauge)*100;
    HeightCalibrationStats(i).InstronError_rel = (HeightCalibrationStats(i).InstronError_abs ./
        HeightCalibrationStats(i).HeightGauge)*100;
end
save([root_path, '\HeightCalStats.mat'],'HeightCalibrationStats');


%% Plotting for Height Calibration Analysis
ScatterplotError_abs = [];
ScatterplotError_rel = [];

for i = 1:size(HeightCalibrationStats,2)
    for j = 1:size(HeightCalibrationStats(i).TopDisplacement,1)
        ScatterplotError_abs = [ScatterplotError_abs;[HeightCalibrationStats(i).HeightGauge,...
                                                      HeightCalibrationStats(i).InstronError_abs
                                                          ,...
                                                      HeightCalibrationStats(i).TopError_abs(j)
                                                          ,...
                                                      HeightCalibrationStats(i).BottomError_abs(j
                                                          )]];

        if not(HeightCalibrationStats(i).HeightGauge ==0)
            ScatterplotError_rel = [ScatterplotError_rel;[HeightCalibrationStats(i).HeightGauge
                ,...
                                                          HeightCalibrationStats(i).
                                                              InstronError_rel,...
                                                          HeightCalibrationStats(i).TopError_rel(
                                                              j),...
                                                          HeightCalibrationStats(i).
                                                              BottomError_rel(j)]];
        end

    end
end

%% Height Calibration Figure
figure()
subplot(1,3,1)
```

```
markersize = 50;
hold on
scatter(ScatterplotError_abs(:,3),ScatterplotError_abs(:,2),markersize,ScatterplotError_abs(:,1)
    ,'filled','MarkerEdgeColor',[0.1 0.1 0.1])
c = colorbar('eastoutside');
c.Label.String = 'Height Gauge Measurement';
title('Absolute Error')
ylabel('Instron Error [mm]')
xlabel('SPECS Error (top platen) [mm]')
axis equal
xlim([-1 1])
ylim([-1 1])
plot([-20,20],[0,0],'k')
plot([0,0],[-20,20],'k')

subplot(1,3,2)
markersize = 50;
hold on
scatter(ScatterplotError_abs(:,4),ScatterplotError_abs(:,2),markersize,ScatterplotError_abs(:,1)
    ,'filled','MarkerEdgeColor',[0.1 0.1 0.1])
c = colorbar('eastoutside');
c.Label.String = 'Height Gauge Measurement';
title('Absolute Error')
ylabel('Instron Error [mm]')
xlabel('SPECS Error (bottom platen) [mm]')
axis equal
xlim([-1 1])
ylim([-1 1])
plot([-20,20],[0,0],'k')
plot([0,0],[-20,20],'k')

subplot(1,3,3)
markersize = 50;
hold on
scatter(ScatterplotError_abs(:,4),ScatterplotError_abs(:,3),markersize,ScatterplotError_abs(:,1)
    ,'filled','MarkerEdgeColor',[0.1 0.1 0.1])
c = colorbar('eastoutside');
c.Label.String = 'Height Gauge Measurement';
title('Absolute Error')
ylabel('SPECS Error (top platen) [mm]')
xlabel('SPECS Error (bottom platen) [mm]')
axis equal
xlim([-1 1])
ylim([-1 1])
plot([-20,20],[0,0],'k')
plot([0,0],[-20,20],'k')

fprintf('\nDone!\n')
toc

%% Individual graphs for height calibration
figure()
markersize = 50;
hold on
scatter(ScatterplotError_abs(:,3),ScatterplotError_abs(:,2),markersize,ScatterplotError_abs(:,1)
    ,'filled','MarkerEdgeColor',[0.1 0.1 0.1])
c = colorbar('eastoutside');
c.Label.String = 'Height Gauge Measurement';
title('Absolute Error')
ylabel('Instron Error [mm]')
xlabel('SPECS Error (top platen) [mm]')
axis equal
xlim([-1 1])
ylim([-1 1])
plot([-20,20],[0,0],'k')
plot([0,0],[-20,20],'k')
```

```
figure()
markersize = 50;
hold on
scatter(ScatterplotError_abs(:,4),ScatterplotError_abs(:,3),markersize,ScatterplotError_abs(:,1)
    ,'filled','MarkerEdgeColor',[0.1 0.1 0.1])
c = colorbar('eastoutside');
c.Label.String = 'Height Gauge Measurement';
title('Absolute Error')
ylabel('SPECS Error (top platen) [mm]')
xlabel('SPECS Error (bottom platen) [mm]')
axis equal
xlim([-1 1])
ylim([-1 1])
plot([-20,20],[0,0],'k')
plot([0,0],[-20,20],'k')

%% Calibration block statistical analysis
fprintf('\n\nCalculating statistics for Calibration Blocks...')
tic
clear CalibrationStats
clear CalibrationBlockError

for i = 1:4 % loop over calibration samples
    DiamCalibrationStats(i).SampleName =  Data_all(CalIndex(i)).SampleName;
    DiamCalibrationStats(i).MeanDiameter = Data_all(CalIndex(i)).Compiled{1:20,'MeanDiameter'};
    DiamCalibrationStats(i).MaxDiameter = Data_all(CalIndex(i)).Compiled{1:20,'MaxDiameter'};
    DiamCalibrationStats(i).MinDiameter = Data_all(CalIndex(i)).Compiled{1:20,'MinDiameter'};
    for j = 1:20
        DiamCalibrationStats(i).StdDev(j,1) = std(Data_all(CalIndex(i)).Compiled{j,'
            SampleDiameter'}{:});
    end
    DiamCalibrationStats(i).MaxStdDev=max(DiamCalibrationStats(i).StdDev);
    DiamCalibrationStats(i).MinStdDev=min(DiamCalibrationStats(i).StdDev);
    DiamCalibrationStats(i).MeanStdDev=mean(DiamCalibrationStats(i).StdDev);
    DiamCalibrationStats(i).MeanDiameterRange=[min(DiamCalibrationStats(i).MeanDiameter),mean(
        DiamCalibrationStats(i).MeanDiameter),max(DiamCalibrationStats(i).MeanDiameter)];
    DiamCalibrationStats(i).StdDevMeanDiameter = std(DiamCalibrationStats(i).MeanDiameter);
    DiamCalibrationStats(i).MaxDiameterRange=[min(DiamCalibrationStats(i).MaxDiameter),mean(
        DiamCalibrationStats(i).MaxDiameter),max(DiamCalibrationStats(i).MaxDiameter)];
    DiamCalibrationStats(i).StdDevMaxDiameter = std(DiamCalibrationStats(i).MaxDiameter);
    DiamCalibrationStats(i).MinDiameterRange=[min(DiamCalibrationStats(i).MinDiameter),mean(
        DiamCalibrationStats(i).MinDiameter),max(DiamCalibrationStats(i).MinDiameter)];
    for j = 1:20 % j = frame, i = sample
        ErrorTMP = Data_all(CalIndex(i)).Compiled{1:20,'SampleDiameter'}{j,1};
        CalibrationBlockError{j,i} = [Data_all(CalIndex(i)).Compiled{1:20,'SampleY'}{j,1},(
            ErrorTMP - ones(size(ErrorTMP)).*NominalDiameter_Cal{i}).*100./NominalDiameter_Cal{i
            }];
    end
end

fprintf('\nDone!\n')
toc
for i = 1:2
    figure()
    hold on
    for j = 1:20
        plot(CalibrationBlockError{j,i}(:,1),CalibrationBlockError{j,i}(:,2),'k.')
    end
    xlabel('Vertical Coordinate [mm]','FontSize',15)
    ylabel('% Relative Error on Diameter','FontSize',15)
    if i==1
        title('Calibration Block 1','FontSize',18);
    else
        title('Calibration Block 3','FontSize',18);
    end
```

```
    %ylim([-1.5 1.5])
end
save([root_path, '\DiamCalStats.mat'],'DiamCalibrationStats');
%%

%% Plotting
%% ------------------------------------------------------------------------%
%--------- Engineering SS Graphs grouped by AR, Colored by Sample
%-------------------------------------------------------------------------%
figure()
hold on
LegendEntries = {};
%Plot all data
for n = 1:size(SampleIndex_50mm)
    i = SampleIndex_50mm(n);
    plot(Data_all(i).Compiled{:,'EngStrain'},Data_all(i).Compiled{:,'EngStress'},'.')
    LegendEntries{n,1} = Data_all(i).SampleName;
end
xlabel('Compressive Engineering Strain');
ylabel('Compressive Engineering Stress [MPa]');
[h,icons,plots,legend_text] = legend(LegendEntries,'Location','SouthEast');
for k = 7:18
    icons(k).MarkerSize = 18;
end
title({'Engineering Stress, Engineering Strain Curve','D_o / H_o = 0.5'})
xlim([-0.05 0.4])


figure()
hold on
LegendEntries = {};
for n = 1:size(SampleIndex_31mm)
    i = SampleIndex_31mm(n);
    plot(Data_all(i).Compiled{:,'EngStrain'},Data_all(i).Compiled{:,'EngStress'},'.')
    LegendEntries{n,1} = Data_all(i).SampleName;
end
xlabel('Compressive Engineering Strain');
ylabel('Compressive Engineering Stress [MPa]');
[h,icons,plots,legend_text] = legend(LegendEntries,'Location','SouthEast');
for k = 7:18
    icons(k).MarkerSize = 18;
end
title({'Engineering Stress, Engineering Strain Curve','D_o / H_o = 0.8'})
xlim([-0.05 0.4])

figure()
hold on
LegendEntries = {};
for n = 1:size(SampleIndex_25mm)
    i = SampleIndex_25mm(n);
    plot(Data_all(i).Compiled{:,'EngStrain'},Data_all(i).Compiled{:,'EngStress'},'.')
    LegendEntries{n,1} = Data_all(i).SampleName;
end
xlabel('Compressive Engineering Strain');
ylabel('Compressive Engineering Stress [MPa]');
[h,icons,plots,legend_text] = legend(LegendEntries,'Location','SouthEast');
for k = 5:12
    icons(k).MarkerSize = 18;
end
title({'Engineering Stress, Engineering Strain Curve','D_o / H_o = 1.0'})
xlim([-0.05 0.4])

%% ------------------------------------------------------------------------%
%--------- Engineering SS Graph Colored by AR
%-------------------------------------------------------------------------%
figure()
```

```
hold on
for n = 1:size(SampleIndex_50mm)
    i = SampleIndex_50mm(n);
    plot(Data_all(i).Compiled{:,'EngStrain'},Data_all(i).Compiled{:,'EngStress'},'r.')
end
for n = 1:size(SampleIndex_31mm)
    i = SampleIndex_31mm(n);
    plot(Data_all(i).Compiled{:,'EngStrain'},Data_all(i).Compiled{:,'EngStress'},'b.')
end
for n = 1:size(SampleIndex_25mm)
    i = SampleIndex_25mm(n);
    plot(Data_all(i).Compiled{:,'EngStrain'},Data_all(i).Compiled{:,'EngStress'},'k.')
end
xlabel('Engineering Strain');
ylabel('Engineering Stress [MPa]');

%% ---------------------------------------------------------------------%
%--------- True SS Graph Colored by AR (Mean Diameter)
%---------------------------------------------------------------------%
figure()
hold on
for n = 1:size(SampleIndex_50mm)
    i = SampleIndex_50mm(n);
    handle_50mm = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MeanDiamTrueStress'},'r.','MarkerSize',4);
end
for n = 1:size(SampleIndex_31mm)
    i = SampleIndex_31mm(n);
    handle_31mm = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MeanDiamTrueStress'},'b.','MarkerSize',4);
end
for n = 1:size(SampleIndex_25mm)
    i = SampleIndex_25mm(n);
    handle_25mm = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MeanDiamTrueStress'},'g.','MarkerSize',4);
end
xlabel('True Compressive Strain')
ylabel('True Compressive Stress [MPa]')
[h,icons,plots,legend_text] = legend([handle_50mm handle_31mm handle_25mm],'H_o = 50mm','H_o = 31
    mm','H_o = 25mm','Location','SouthEast');
for k = 4:9
    icons(k).MarkerSize = 12;
end
title({'True Stress, True Strain Curve','Median Diameter All Samples'})

%% ---------------------------------------------------------------------%
%--------- True SS Graph Colored by Diameter Used Do/Ho = 0.5
%---------------------------------------------------------------------%
figure()
hold on
for n = 1:size(SampleIndex_50mm)
    i = SampleIndex_50mm(n);
    handle_min = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MinDiamTrueStress'},'r.','MarkerSize',4);
    handle_x_min = plot(Data_all(i).Compiled{end-1,'TrueStrain'},Data_all(i).Compiled{end-1,'
        MinDiamTrueStress'},'kx','MarkerSize',8,'LineWidth',1);
end

for n = 1:size(SampleIndex_50mm)
    i = SampleIndex_50mm(n);
    handle_mean = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MeanDiamTrueStress'},'b.','MarkerSize',4);
    handle_x_mean = plot(Data_all(i).Compiled{end-1,'TrueStrain'},Data_all(i).Compiled{end-1,'
        MeanDiamTrueStress'},'kx','MarkerSize',8,'LineWidth',1);
end
```

```
for n = 1:size(SampleIndex_50mm)
    i = SampleIndex_50mm(n);
    handle_max = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MaxDiamTrueStress'},'g.','MarkerSize',4);
    handle_x_max = plot(Data_all(i).Compiled{end-1,'TrueStrain'},Data_all(i).Compiled{end-1,'
        MaxDiamTrueStress'},'kx','MarkerSize',8,'LineWidth',1);
end
handle_Bao = plot(-Data_Bao(1).Compiled{:,'TrueStrain'},Data_Bao(1).Compiled{:,'
    MeanDiamTrueStress'},'k:','LineWidth',2);
handle_Khan = plot(Data_Khan(:,1),Data_Khan(:,2),'k--','LineWidth',1);



xlim([0 0.5])
ylim([0 800])
xlabel('True Compressive Strain','FontSize',12)
ylabel('True Compressive Stress [MPa]','FontSize',12)
[h,icons,plots,legend_text] = legend([handle_min handle_mean handle_max handle_x_min handle_Bao
    handle_Khan],'Minimum Diameter','Mean Diameter','Max Diameter','Sample Fracture','Weirzbicki
    & Bao Results','Khan and Liu Results','Location','SouthEast');
for k = 7:12
    icons(k).MarkerSize = 16;
end
title({'True Stress, True Strain Curve','D_o / H_o = 0.5'})


%% ------------------------------------------------------------------------%
%--------- True SS Graph Colored by Diameter Used Do/Ho = 0.8
%-------------------------------------------------------------------------%
figure()
hold on
for n = 1:size(SampleIndex_31mm)
    i = SampleIndex_31mm(n);
    handle_min = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MinDiamTrueStress'},'r.','MarkerSize',4);
    handle_x_min = plot(Data_all(i).Compiled{end-1,'TrueStrain'},Data_all(i).Compiled{end-1,'
        MinDiamTrueStress'},'kx','MarkerSize',8,'LineWidth',1);
end

for n = 1:size(SampleIndex_31mm)
    i = SampleIndex_31mm(n);
    handle_mean = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MeanDiamTrueStress'},'b.','MarkerSize',4);
    handle_x_mean = plot(Data_all(i).Compiled{end-1,'TrueStrain'},Data_all(i).Compiled{end-1,'
        MeanDiamTrueStress'},'kx','MarkerSize',8,'LineWidth',1);
end

for n = 1:size(SampleIndex_31mm)
    i = SampleIndex_31mm(n);
    handle_max = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MaxDiamTrueStress'},'g.','MarkerSize',4);
    handle_x_max = plot(Data_all(i).Compiled{end-1,'TrueStrain'},Data_all(i).Compiled{end-1,'
        MaxDiamTrueStress'},'kx','MarkerSize',8,'LineWidth',1);
end
handle_Bao = plot(-Data_Bao(1).Compiled{:,'TrueStrain'},Data_Bao(1).Compiled{:,'
    MeanDiamTrueStress'},'k:','LineWidth',2);
handle_Khan = plot(Data_Khan(:,1),Data_Khan(:,2),'k--','LineWidth',1);



xlim([0 0.5])
ylim([0 800])
xlabel('True Compressive Strain','FontSize',12)
ylabel('True Compressive Stress [MPa]','FontSize',12)
```

```
[h,icons,plots,legend_text] = legend([handle_min handle_mean handle_max handle_x_min handle_Bao
    handle_Khan],'Minimum Diameter','Mean Diameter','Max Diameter','Sample Fracture','Weirzbicki
    & Bao Results','Khan and Liu Results','Location','SouthEast');
for k = 7:12
    icons(k).MarkerSize = 16;
end
title({'True Stress, True Strain Curve','D_o / H_o = 0.8'})


%% ------------------------------------------------------------------------%
%---------- True SS Graph Colored by Diameter Used Do/Ho = 1.0
%------------------------------------------------------------------------%
figure()
hold on
for n = 1:size(SampleIndex_25mm)
    i = SampleIndex_25mm(n);
    handle_min = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MinDiamTrueStress'},'r.','MarkerSize',4);
    handle_x_min = plot(Data_all(i).Compiled{end-1,'TrueStrain'},Data_all(i).Compiled{end-1,'
        MinDiamTrueStress'},'kx','MarkerSize',8,'LineWidth',1);
end

for n = 1:size(SampleIndex_25mm)
    i = SampleIndex_25mm(n);
    handle_mean = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MeanDiamTrueStress'},'b.','MarkerSize',4);
    handle_x_mean = plot(Data_all(i).Compiled{end-1,'TrueStrain'},Data_all(i).Compiled{end-1,'
        MeanDiamTrueStress'},'kx','MarkerSize',8,'LineWidth',1);
end

for n = 1:size(SampleIndex_25mm)
    i = SampleIndex_25mm(n);
    handle_max = plot(Data_all(i).Compiled{:,'TrueStrain'},Data_all(i).Compiled{:,'
        MaxDiamTrueStress'},'g.','MarkerSize',4);
    handle_x_max = plot(Data_all(i).Compiled{end-1,'TrueStrain'},Data_all(i).Compiled{end-1,'
        MaxDiamTrueStress'},'kx','MarkerSize',8,'LineWidth',1);
end

handle_Bao = plot(-Data_Bao(1).Compiled{:,'TrueStrain'},Data_Bao(1).Compiled{:,'
    MeanDiamTrueStress'},'k:','LineWidth',2);
handle_Khan = plot(Data_Khan(:,1),Data_Khan(:,2),'k--','LineWidth',1);



xlim([0 0.5])
ylim([0 800])
xlabel('True Compressive Strain','FontSize',12)
ylabel('True Compressive Stress [MPa]','FontSize',12)
[h,icons,plots,legend_text] = legend([handle_min handle_mean handle_max handle_x_min handle_Bao
    handle_Khan],'Minimum Diameter','Mean Diameter','Max Diameter','Sample Fracture','Weirzbicki
    & Bao Results','Khan and Liu Results','Location','SouthEast');
for k = 7:12
    icons(k).MarkerSize = 16;
end
title({'True Stress, True Strain Curve','D_o / H_o = 1.0'})

%% ------------------------------------------------------------------------%
%---------- Create Force-Displacement Curves
%------------------------------------------------------------------------%
figure()
hold on
for n = 1:size(SampleIndex_25mm)
    i = SampleIndex_25mm(n);
```

```
    plot(-Data_all(i).Compiled{:,'SampleDisplacement'},Data_all(i).Compiled{:,'Load'}/1000,'.','
        MarkerSize',4);
end
xlim([0 10])
ylim([0 600])
legend(Samples_25mm,'Location','NorthWest')
ylabel('Force [kN]')
xlabel('Displacement [mm]')

figure()
hold on
for n = 1:size(SampleIndex_31mm)
    i = SampleIndex_31mm(n);
    plot(-Data_all(i).Compiled{:,'SampleDisplacement'},Data_all(i).Compiled{:,'Load'}/1000,'.','
        MarkerSize',4);
end
xlim([0 12])
ylim([0 600])
legend(Samples_31mm,'Location','NorthWest')
ylabel('Force [kN]')
xlabel('Displacement [mm]')

figure()
hold on
for n = 1:size(SampleIndex_50mm)
    i = SampleIndex_50mm(n);
    plot(-Data_all(i).Compiled{:,'SampleDisplacement'},Data_all(i).Compiled{:,'Load'}/1000,'.','
        MarkerSize',4);
end
xlim([0 20])
ylim([0 600])
legend(Samples_50mm,'Location','NorthWest')
ylabel('Force [kN]')
xlabel('Displacement [mm]')

figure()
hold on
for n = 1:size(SampleIndex_25mm)
    i = SampleIndex_25mm(n);
    handle_25mm=plot(Data_all(i).Compiled{:,'Load'}/1000,-Data_all(i).Compiled{:,'
        SampleDisplacement'},'r.','MarkerSize',4);
end
for n = 1:size(SampleIndex_31mm)
    i = SampleIndex_31mm(n);
    handle_31mm=plot(Data_all(i).Compiled{:,'Load'}/1000,-Data_all(i).Compiled{:,'
        SampleDisplacement'},'b.','MarkerSize',4);
end
for n = 1:size(SampleIndex_50mm)
    i = SampleIndex_50mm(n);
    handle_50mm=plot(Data_all(i).Compiled{:,'Load'}/1000,-Data_all(i).Compiled{:,'
        SampleDisplacement'},'k.','MarkerSize',4);
end
[h,icons,plots,legend_text] = legend([handle_50mm handle_31mm handle_25mm],{'H_o = 50mm','H_o =
    31mm','H_o = 25mm'},'Location','NorthWest','FontSize',11);
for k = [5,7,9]
    icons(k).MarkerSize = 16;
end
ylim([0 20])
xlabel('Force [kN]','FontSize',14)
ylabel('Displacement [mm]','FontSize',14)

%% ------------------------------------------------------------------------%
%--------- Plot Error Envelope
%-------------------------------------------------------------------------%
```

```
Sample14 = [Data_all(16).Compiled{:,{'SampleHeight','TrueStrain','MeanDiamTrueStress'}}];%True
    Stress/Strain data for Sample 14
Sample14 = sortrows(Sample14,2); %sort data by strain for smoother display as envelope
Sample14 = Sample14(Sample14(:,2)>=0,:); %remove samples with negative strains (outliers)
Sample14 = Sample14(1:10:size(Sample14,1),:); % take every 10th data point.
Sample14 = Sample14(1:size(Sample14,1)-1,:); % Last point (instant following fracture) is removed
e_strain = sqrt((1./(sqrt(2).*Sample14(:,1))).^2+(1./(sqrt(2).*Sample14(1,1))).^2);
e_stress = 0.0491.*Sample14(:,3);
Envelope_Lower = [Sample14(:,2)+e_strain Sample14(:,3)-e_stress];
Envelope_Upper = [Sample14(:,2)-e_strain Sample14(:,3)+e_stress];

figure()
hold on

plot(Envelope_Upper(:,1),Envelope_Upper(:,2),'b','LineWidth',2)
plot(Sample14(:,2),Sample14(:,3),'k','LineWidth',2)
plot(Envelope_Lower(:,1),Envelope_Lower(:,2),'r','LineWidth',2)
ylim([0 800])
legend('Upper Envelope Upper Bound','Sample 14 Smoothed Data (Mean Diameter)','Error Envelope
    Lower Bound','Location','SouthEast');
xlabel('True Compressive Strain','FontSize',12)
ylabel('True Compressive Stress [MPa]','FontSize',12)

%% ------------------------------------------------------------------------%
%--------- Save Data to Output File
%------------------------------------------------------------------------%
if SaveOutput ==1
    tic
        fprintf('\n saving...\n')
        save([root_path,'\FullyProcessedData.mat'],'Data','-v7.3')
    toc
end
```

Functions

Postprocessing__DataParse_V2_0

```
function Data = Postprocessing__DataParse_V2_0(root_input_path,filename_input,root_output_path,
    filename_output,Flag_OpenFiles,Flag_Instron,Flag_Diameter)

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Postprocessing__DataParse_V2_0.m
%
%   This function is designed to read data from a batch of SPECS data folders,
%   read in the raw data, and do the time consuming data processing step
%   to convert edge detections to diameter measurements.
%
%
%   Note that there is a double underscore, to keep all functions together
%   and separate from the main script.
%
%               Author: Andrea Felling
%   Last Edit: March 28, 2016
%   Edited By: Andrea Felling
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% User Variables
ThresholdFSR =    3; %Threshold (Volts) for FSR signal to count as 'on'
CenterlineX  = 2040; %Vertical line (px) to separate 'left' and 'right' sides
CenterlineY  =  875; %Horizontal line (px) to separate 'top' and 'bottom' sides
CircleRadius = [30 70]; %Range of acceptable circle radius values (px)
ThresholdCircle = 500; %Minimum circle detection score for a circle to be accepted
ThresholdTopPlaten = 11; % Number of mm between top platen detection and start of edges
ThresholdBottomPlaten = 11; % Number of mm between bottom platen detection and start of edges
ThresholdEdgePairing = 1;
CircleROI = [ ];
%% Load Parsed and Sorted Data
%root_path = '..\..\03 - SPECS Data\AL2024 Samples';
data_path = [root_input_path,'\',filename_input];

% reading from files takes a couple of minutes, simply re-load if this step
% has already been performed to save time
if exist(data_path,'file') && (Flag_OpenFiles == 0)
    fprintf('RawData.mat found...\nLoading Data...')
    load(data_path);
    fprintf('\nDone!\n\n')
else
    fprintf('Running "Postprocessing__OpenFiles_V1_0.m"\n\n')
    Data = Postprocessing__OpenFiles_V1_0(root_input_path);
end

%% Data Processing
%
DataOut = struct([]);
Progressbar_all = waitbar(0,'Overall Progress:');
for n = 1:size(Data,2) %Loop through all files
    waitbar(n/size(Data,2),Progressbar_all)
    tic
    fprintf('Processing "%s"...\n',Data(n).SampleName)

%-------------------------------------------------------------------------%
    fprintf('Aligning timestamps...\n')
    FrameTimingData = FramesFromStrobe(Data(n).Strobe, Data(n).Recording);
    if Flag_Instron ==1
        InstronStart = Postprocessing__AlignInstronData_V1_0(Data(n).FSR,ThresholdFSR);
        FrameTimingData = FrameTimingData-InstronStart;
```

```
    else
        FrameTimingData = FrameTimingData - FrameTimingData(1,1);
    end
    Data(n).Frames = table(Data(n).Recording{:,'Frame'},FrameTimingData(:,1),FrameTimingData(:,2)
        ,'VariableNames',{'FrameNumber','StartTime','EndTime'});
%-------------------------------------------------------------------------%
    fprintf('Averaging load for each frame...\n')
    % Pull out instron data for each frame, then average the load to get
    % the frame's load
    if Flag_Instron ==1
        for i = 1:size(Data(n).Frames,1)
            InstronRows = (Data(n).Instron{:,'Time_s'}>Data(n).Frames{i,'StartTime'})&(Data(n).
                Instron{:,'Time_s'}<Data(n).Frames{i,'EndTime'});
            Data(n).Load(i,:) = table(Data(n).Frames{i,'FrameNumber'},mean(Data(n).Instron{
                InstronRows,'Load_N'}));
        end
    else
        Data(n).Load = table(Data(n).Frames{:,'FrameNumber'},zeros(size(Data(n).Frames{:,'
            FrameNumber'})));
    end
    Data(n).Load.Properties.VariableNames = {'FrameNumber','Load_N'};
%-------------------------------------------------------------------------%
    fprintf('Analyzing platen position for each frame...\n')
    % Sort through circle data for each frame to get the top and bottom
    % platen positions, displacement, and sample height
    ValidCircles = (Data(n).Circles{:,'CircleRadius_px'} > CircleRadius(1,1)) & ...
                   (Data(n).Circles{:,'CircleRadius_px'} < CircleRadius(1,2)) & ...
                   (Data(n).Circles{:,'CircleDetectionScore'} > ThresholdCircle(1,1)) & ...
                   (Data(n).Circles{:,'X_px'} > Data(n).Settings.ROI{1,'Left'}) & ...
                   (Data(n).Circles{:,'X_px'} < Data(n).Settings.ROI{1,'Right'}) & ...
                   (Data(n).Circles{:,'Y_px'} > Data(n).Settings.ROI{1,'Top'}) & ...
                   (Data(n).Circles{:,'Y_px'} < Data(n).Settings.ROI{1,'Bottom'});
    Data(n).Circles = Data(n).Circles(ValidCircles,:);
    for i = 1:size(Data(n).Frames,1)
        CircleRows = (Data(n).Circles{:,'Frame'}==Data(n).Frames{i,'FrameNumber'});
        CircleData = Data(n).Circles(CircleRows,:);
        CenterlineX = mean(CircleData{:,'X_px'});
        CenterlineY = mean(CircleData{:,'Y_px'});
        PlatenTL = CircleData((CircleData{:,'Y_px'}<CenterlineY)&(CircleData{:,'X_px'}<
            CenterlineX),:);
        PlatenTR = CircleData((CircleData{:,'Y_px'}<CenterlineY)&(CircleData{:,'X_px'}>
            CenterlineX),:);
        PlatenBL = CircleData((CircleData{:,'Y_px'}>CenterlineY)&(CircleData{:,'X_px'}<
            CenterlineX),:);
        PlatenBR = CircleData((CircleData{:,'Y_px'}>CenterlineY)&(CircleData{:,'X_px'}>
            CenterlineX),:);
        Data(n).AvgCircles(i,:) = table(Data(n).Frames{i,'FrameNumber'},...
                                        mean(PlatenTL{:,'X_mm'}),mean(PlatenTL{:,'Y_mm'}),...
                                        mean(PlatenTR{:,'X_mm'}),mean(PlatenTR{:,'Y_mm'}),...
                                        mean(PlatenBL{:,'X_mm'}),mean(PlatenBL{:,'Y_mm'}),...
                                        mean(PlatenBR{:,'X_mm'}),mean(PlatenBR{:,'Y_mm'}));
    end
    Data(n).AvgCircles.Properties.VariableNames = {'FrameNumber',...
                                                   'TopLeft_X_mm','TopLeft_Y_mm',...
                                                   'TopRight_X_mm','TopRight_Y_mm',...
                                                   'BottomLeft_X_mm','BottomLeft_Y_mm',...
                                                   'BottomRight_X_mm','BottomRight_Y_mm'};

    % Take mean height of both platens  (nanmean function ignores NaN values, which may be
        present with missed detections)
    Data(n).Platens = table(Data(n).Frames{:,'FrameNumber'}, ...
                            nanmean([Data(n).AvgCircles{:,'TopLeft_Y_mm'} Data(n).AvgCircles
                                {:,'TopRight_Y_mm'}],2),...
                            nanmean([Data(n).AvgCircles{:,'BottomLeft_Y_mm'} Data(n).AvgCircles
                                {:,'BottomRight_Y_mm'}],2));
    Data(n).Platens.Properties.VariableNames =  {'FrameNumber','TopPlaten_mm','BottomPlaten_mm'};
```

```
%-------------------------------------------------------------------------%
    fprintf('Isolating Sample Edge Detections...\n')
    for i = 1:size(Data(n).Frames,1)
        AllEdges = (Data(n).Edges{:,'Frame'}==Data(n).Frames{i,'FrameNumber'}) & ...
                   (Data(n).Edges{:,'X_px'} > Data(n).Settings.ROI{1,'Left'}) & ...
                   (Data(n).Edges{:,'X_px'} < Data(n).Settings.ROI{1,'Right'}) & ...
                   (Data(n).Edges{:,'Y_px'} > Data(n).Settings.ROI{1,'Top'}) & ...
                   (Data(n).Edges{:,'Y_px'} < Data(n).Settings.ROI{1,'Bottom'});

        CenterlineX = (Data(n).Settings.ROI{1,'Left'}+Data(n).Settings.ROI{1,'Right'})/2;

        LeftEdge = AllEdges & Data(n).Edges{:,'X_px'} < CenterlineX;
        RightEdge = AllEdges & Data(n).Edges{:,'X_px'} > CenterlineX;
        Data(n).LeftEdge(i,:) = table((Data(n).Frames{i,'FrameNumber'}),...
                                      {Data(n).Edges{LeftEdge,2}},...
                                      {Data(n).Edges{LeftEdge,3}},...
                                      {Data(n).Edges{LeftEdge,4}},...
                                      {Data(n).Edges{LeftEdge,5}});

        Data(n).RightEdge(i,:) = table((Data(n).Frames{i,'FrameNumber'}),...
                                      {Data(n).Edges{RightEdge,2}},...
                                      {Data(n).Edges{RightEdge,3}},...
                                      {Data(n).Edges{RightEdge,4}},...
                                      {Data(n).Edges{RightEdge,5}});


    end
    Data(n).LeftEdge.Properties.VariableNames = {'FrameNumber','X_px','Y_px','X_mm','Y_mm'};
    Data(n).RightEdge.Properties.VariableNames = {'FrameNumber','X_px','Y_px','X_mm','Y_mm'};
%-------------------------------------------------------------------------%
    fprintf('Calculating Diameter (Can take 15-30 minutes) ...\n')
    if Flag_Diameter == 1
        for i = 1:size(Data(n).Frames,1)
            j = 1;
            diameter = [];
            for searchline = Data(n).Settings.ROI{1,'Top'}:ThresholdEdgePairing:Data(n).Settings.
                ROI{1,'Bottom'}

                searchline_lower = searchline + ThresholdEdgePairing;
                searchline_upper = searchline - ThresholdEdgePairing;
                leftline = (Data(n).LeftEdge{i,'Y_px'}{:} > searchline_upper) & ...
                           (Data(n).LeftEdge{i,'Y_px'}{:} < searchline_lower);
                rightline = (Data(n).RightEdge{i,'Y_px'}{:} > searchline_upper) & ...
                            (Data(n).RightEdge{i,'Y_px'}{:} < searchline_lower);

                leftX = Data(n).LeftEdge{i,'X_mm'}{:}(leftline);
                leftY = Data(n).LeftEdge{i,'Y_mm'}{:}(leftline);
                rightX = Data(n).RightEdge{i,'X_mm'}{:}(rightline);
                rightY = Data(n).RightEdge{i,'Y_mm'}{:}(rightline);

                if isempty(leftX)==0 && isempty(rightX)==0
                    leftX_avg = mean(leftX,1);
                    leftY_avg = mean(leftY,1);
                    rightX_avg = mean(rightX,1);
                    rightY_avg = mean(rightY,1);
                    diameter(j,:) = [(leftY_avg+rightY_avg)/2, leftX_avg, rightX_avg, rightX_avg-
                        leftX_avg];
                    j = j+1;

                end

            end
            if isempty(diameter)==0
                Data(n).Diameter(i,:) = table(Data(n).Frames{i,'FrameNumber'},...
                                              {diameter(:,1)},{diameter(:,2)},...
                                              {diameter(:,3)},{diameter(:,4)});
```

```
            else
%               Data(n).Diameter(i,:) = table(Data(n).Frames{i,'FrameNumber'},...
%                                             {[]},{[]},...
%                                             {[]},{[]});
            end
        end
    else
        diameter = double([1,1,1,1;2,2,2,2]);
        for i = 1:size(Data(n).Frames,1)

            Data(n).Diameter(i,:) = table(Data(n).Frames{i,'FrameNumber'},...
                                          {diameter(:,1)},{diameter(:,2)},...
                                          {diameter(:,3)},{diameter(:,4)});
        end
    end

    Data(n).Diameter.Properties.VariableNames = {'FrameNumber','Height_mm','Left_mm','Right_mm','
        Diameter_mm'};

%
%     indices = 1:40:400;
%     for i = indices
%         figure()
%         hold on
%         plot(Data(4).LeftEdge{i,'X_mm'}{:},-Data(4).LeftEdge{i,'Y_mm'}{:},'r.')
%         plot(Data(4).RightEdge{i,'X_mm'}{:},-Data(4).RightEdge{i,'Y_mm'}{:},'b.')
%         axis equal
%         xlim([(min(Data(4).LeftEdge{i,'X_mm'}{:})-20),(max(Data(4).RightEdge{i,'X_mm'}{:})+20)
    ]);
%     end


%-------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------%
%-------------------------------------------------------------------------------%
    fprintf('Done!\n')
    toc
    fprintf('\n\n')


    DataOut(n).SampleName = Data(n).SampleName;
    DataOut(n).Settings = Data(n).Settings;
    DataOut(n).Frames = Data(n).Frames;
    DataOut(n).Load = Data(n).Load;
    DataOut(n).Circles = Data(n).AvgCircles;
    DataOut(n).Platens = Data(n).Platens;
    DataOut(n).LeftEdge = Data(n).LeftEdge;
    DataOut(n).RightEdge = Data(n).RightEdge;
    DataOut(n).Diameter = Data(n).Diameter;


end
save([root_output_path,'\',filename_output],'DataOut','-v7.3');
```

Postprocessing__OpenFiles_V1_0

```
function Data = Postprocessing__OpenFiles_V1_0(root_path)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                                   OpenFiles.m
%
%
% This function opens all sample data in the selected folder by reading
% data directly from LabVIEW output text files
%
% See Postprocessing_RawDataParse for details on file structure conventions
%
%   Author: Andrea Felling
%   Last Edit: May 31, 2016
%   Edited By: Andrea Felling
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% User Setup (to be delected when OpenFiles becomes a function)
% close all
% clear all
% clc
% tic
% root_path = '..\..\03 - SPECS Data\AL2024 Samples';

%% Directory Structure Operations to Obtain Data File Paths
folder_names = dir(root_path); % list contents of root path

% Parse Folder Names down to Folders of Interest
for i = size(folder_names,1):-1:1; % loop through folders and files found
    if folder_names(i).isdir == 1 % check if the item is a folder
        % Create logical string that identifies each character as
        % alphanumeric or not.  A folder must contain alphanumeric
        % characters to be valid.  '.' and '..' type filenames that get
        % created by the system when folders are created and deleted will
        % only appear as arrays of 0's
        nametest = isstrprop(folder_names(i).name,'alphanum');
        valid_folder_name = sum(nametest);
        if valid_folder_name == 0
            folder_names(i)=[];
        end
    else
        folder_names(i)=[];
    end
end

% Loop Through Folder Names and get Sample Names, Instron Data paths, and
% Containing Folder paths
DataPaths=struct([]);
for i = size(folder_names,1):-1:1
    sample_names = dir([root_path,'\',folder_names(i).name]); % if instron data does not exist,
         parse by using strcomp to throw out calibration dots, background images and junk folders
    for j = size(sample_names,1):-1:1
        if (not(isempty(strfind(sample_names(j,1).name, 'junk')))) ||... % = 1 if filename
             contains 'junk'
            (not(isempty(strfind(sample_names(j,1).name, 'Calibration')))) ||... % = 1 if filename
                 contains 'Calibration'
            (not(isempty(strfind(sample_names(j,1).name, 'Background')))) ||... % = 1 if filename
                 contains 'Background'
            (not(isempty(strfind(sample_names(j,1).name, 'Instron')))) ||... % = 1 if filename
                 contains 'Background'
            (sample_names(j,1).isdir == 0)||...
            (strncmp(sample_names(j,1).name,'.',1)) % = 1 if the first character in the filename
                 is '.', denoting a hidden system file
```

```
            sample_names(j) = []; % delete file from sample_names variable if any of the above
                conditions are met

        end

    end

    for j = 1:size(sample_names,1)

        n = size(DataPaths,1)+1;
        DataPaths(n,1).SampleName = sample_names(j).name(1:end-17);
        DataPaths(n,1).LabviewFolder = [root_path,'\',folder_names(i).name];
        DataPaths(n,1).Instron = [root_path,'\',folder_names(i).name,'\','Instron Data\',
            DataPaths(n,1).SampleName,'.csv'];
        DataPaths(n,1).LabviewSubfolder = [root_path,'\',folder_names(i).name,'\',sample_names(j)
            .name];
        DataPaths(n,1).FSR = [DataPaths(n,1).LabviewSubfolder,'\','load.out'];
        DataPaths(n,1).Recording = [DataPaths(n,1).LabviewSubfolder,'\','recording.out'];
        DataPaths(n,1).Strobe = [DataPaths(n,1).LabviewSubfolder,'\','strobe.out'];
        DataPaths(n,1).Settings = [DataPaths(n,1).LabviewSubfolder,'\ProcessedData\','
            settings_postprocessing.out'];
        DataPaths(n,1).SearchLines = [DataPaths(n,1).LabviewSubfolder,'\ProcessedData\','
            searchlines.out'];
        DataPaths(n,1).Edges = [DataPaths(n,1).LabviewSubfolder,'\ProcessedData\','edges.out'];
        DataPaths(n,1).Circles = [DataPaths(n,1).LabviewSubfolder,'\ProcessedData\','circles.out
            '];

    end

end

%% Open Data Files Into Data Structure
for n = 1:size(DataPaths,1)

    Data(n).SampleName = DataPaths(n).SampleName;

    % Open Button-press signal from Force Sensing Resistor
    fid = fopen(DataPaths(n).FSR);
        FSR = textscan(fid, '%s %s %f','HeaderLines',4);
        FSR{:,2} = datenum(FSR{:,2},'HH:MM:SS.FFF')*24*60*60;
        Data(n).FSR = table(FSR{:,2:3},'VariableNames',{'Time_s','Voltage'});
    fcid = fclose(fid);


    % Open Recording on/off boolean from LabVIEW VI
    fid = fopen(DataPaths(n).Recording);
        Recording = textscan(fid, '%u %u');
        Data(n).Recording = table(Recording{:,1:2},'VariableNames',{'Frame','Boolean'});
    fcid = fclose(fid);


    % Open Strobe signal from Camera
    fid = fopen(DataPaths(n).Strobe);
        Strobe = textscan(fid, '%s %s %u','HeaderLines',5);
        Strobe{:,2} = datenum(Strobe{:,2},'HH:MM:SS.FFF')*24*60*60;
        Data(n).Strobe = table(Strobe{:,2:3},'VariableNames',{'Time_s','Strobe'});
    fcid = fclose(fid);

    % Load Settings from LabVIEW output
    fid = fopen(DataPaths(n).Settings);
        tline = fgetl(fid);
        while ischar(tline)
            if strcmp('-- Particle Filter Settings: Calibration Dots --',tline)
                tline=fgetl(fid);
                CalDots = [];
```

```
            while strcmp('----------------------------------------------------',tline)==0
                tline=fgetl(fid);
                CalibrationDots = textscan(tline,'%u%f%f%u%u');
                CalDots = [CalDots;CalibrationDots{1,1:3}];
            end
            Data(n).Settings.CalDots = table(CalDots(:,1),CalDots(:,2),CalDots(:,3),'
                VariableNames',{'Parameter','MinValue','MaxValue'});
        end
        if strcmp('---------- Calibration Grid Settings ----------',tline)
            tline=fgetl(fid);
            tline=fgetl(fid);
            CalGrid = textscan(tline,'%f%f%u');
            Data(n).Settings.CalGrid = table(CalGrid{:,:},'VariableNames',{'XSpacing','
                YSpacing','Unit'});
        end
        if strcmp('---------- ROI ----------',tline)
            tline=fgetl(fid);
            tline=fgetl(fid);
            ROI = textscan(tline,'%u%u%u%u');
            Data(n).Settings.ROI = table(ROI{:,:},'VariableNames',{'Left','Top','Right','
                Bottom'});
        end
        if strcmp('---------- Calibration Error Stats ----------',tline)
            tline=fgetl(fid);
            tline=fgetl(fid);
            CalError = textscan(tline,'%f%f%f%f');
            Data(n).Settings.CalError = table(CalError{:,:},'VariableNames',{'MeanError','
                MaxError','StandardDeviation','PercentDistortion'});
        end
        if strcmp('---------- Binary Threshold ----------',tline)
            tline=fgetl(fid);
            BinaryThreshold = textscan(tline,'%u');
            Data(n).Settings.BinaryThreshold = BinaryThreshold{1,1};
        end
        if strcmp('---------- Circle Detection Settings ----------',tline)
            tline=fgetl(fid);
            tline=fgetl(fid);
            CircleDetect = textscan(tline,'%u%u%u%u%u%u%u%u%u');
            Data(n).Settings.CircleEdgeExtraction = table(CircleDetect{:,:},'VariableNames
                ',{'ExtractionMode','EdgeThreshold','EdgeFilterSize','MinLength','
                RowSearchStepSize','ColSearchStepSize','MaxEndpointGap','Closed','Subpixel'})
                ;
            tline=fgetl(fid);
            tline=fgetl(fid);
            tline=fgetl(fid);
            CircleDetect = textscan(tline,'%f%f%f%u%u%u');
            Data(n).Settings.CircleMatchParameters = table(CircleDetect{:,:},{'
                MinimumMatchScore','MinimumRadius','MaximumRadius','Rotation','Scale','
                Occlusion'});
            tline=fgetl(fid);
            tline=fgetl(fid);
            tline=fgetl(fid);
            CircleConstraints = [];
            while ischar(tline)
                CircleDetect = textscan(tline,'%u%f%f');
                CircleConstraints = [CircleConstraints;CircleDetect{1,1:3}];
                tline=fgetl(fid);
            end
            Data(n).Settings.CircleConstraints = table(CircleConstraints(:,1),
                CircleConstraints(:,2),CircleConstraints(:,3),'VariableNames',{'Parameter','
                MinValue','MaxValue'});
        end

        tline = fgetl(fid);
    end
fcid = fclose(fid);
```

```
    % Open search lines from LabVIEW output
    fid = fopen(DataPaths(n).SearchLines);
        SearchLines = textscan(fid, '%f%f%f%f%f','HeaderLines',1);
        Data(n).SearchLines = table(SearchLines{:,1:5},'VariableNames',{'Frame','XStart_px','
            YStart_px','XEnd_px','YEnd_px'});
        Data(n).SearchLines.Properties.VariableUnits = {'' 'px' 'px' 'px' 'px'};
    fcid = fclose(fid);

    % Open edge detections from LabVIEW output
    fid = fopen(DataPaths(n).Edges);
        Edges = textscan(fid, '%f%f%f%f%f','HeaderLines',1);
        Data(n).Edges = table(Edges{:,1:5},'VariableNames',{'Frame','X_px','Y_px','X_mm','Y_mm'})
            ;
    fcid = fclose(fid);

    % Open circle detections from LabVIEW output
    fid = fopen(DataPaths(n).Circles);
        Circles = textscan(fid, '%f%f%f%f%f%f%f','HeaderLines',1);
        Data(n).Circles = table(Circles{:,1:7},'VariableNames',{'Frame','X_px','Y_px','X_mm','
            Y_mm','CircleRadius_px','CircleDetectionScore'});
    fcid = fclose(fid);

    % Open Instron data files
    % IF THIS SECTION IS NOT WORKING, YOU MAY NEED TO CONVERT YOUR FILE(s)
    % TO ANSI FORMATTING
    if exist(DataPaths(n).Instron,'file')
        fid = fopen(DataPaths(n).Instron);
            Instron = textscan(fid, '%f%f%f%f%s','Delimiter',',','HeaderLines',3);
            Data(n).Instron = table(Instron{:,1:5},'VariableNames',{'Time_s','Stress_MPa','
                Position_mm','Load_N','Zone'});
            Data(n).Instron{:,'Time_s'} = Data(n).Instron{:,'Time_s'}.*60; % Data imports in
                minutes, convert to seconds
            Data(n).Instron{:,'Load_N'} = Data(n).Instron{:,'Load_N'}.*1000; % Data imports in kN
                , convert to N
        fcid = fclose(fid);
    else
            placeholder1 = [1,2,3]';
            placeholder2 = [0,0,0]';
            placeholder3 = [{'Placeholder'},{'Placeholder'},{'Placeholder'}]';
            Data(n).Instron = table(placeholder1,placeholder2,placeholder2,placeholder2,
                placeholder3,'VariableNames',{'Time_s','Stress_MPa','Position_mm','Load_N','Zone
                '});
    end
end
save([root_path,'\RawData.mat'],'Data','-v7.3')
return
```

Postprocessing__AlignInstronData_V1_0

```
function InstronStartTime = Postprocessing__AlignInstronData_V1_0(FSR,ThresholdFSR)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%             Postprocessing__AlignInstronData_V1_0.m
%
%   This function takes input data from a Force Sensing Resistor (FSR) and
%   a voltage threshold, and calculates the timestamp which corresponds to
%   peak FSR voltage
%
%   This timestamp corresponds to the start of Instron testing, and can be
%   used to align Instron timestamps to DAQ timestamps
%
%   Author: Andrea Felling
%   Last Edit: May 31, 2016
%   Edited By: Andrea Felling
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

FSR = FSR(FSR{:,'Voltage'}>ThresholdFSR,:);
[~,locs,w,p] = findpeaks(FSR{:,2});
peakscore = w.*p;
peaks = locs(peakscore>1);
InstronStartTime = FSR{peaks(end),1};
figure()
plot(FSR{:,'Time_s'},FSR{:,'Voltage'});
hold on
plot(InstronStartTime,max(FSR{:,2}),'rx');

end
```

Postprocessing__FramesFromStrobe_V1_0

```
function [FrameData] = Postprocessing__FramesFromStrobe_V1_0(StrobeData, RecordingData)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function takes a path to a strobe file as input, and outputs a
% matrix of the following form:
%       Frame #          Start Timestamp         End Timestamp
%           1           1471939.507242419      1471939.507246470
%           2           1471939.507283446      1471939.507289498
%           :                   :                      :
%           :                   :                      :
%
%
%
%
%   Written:    JUN 10 2015
%   Author:     ANDREA FELLING
%
%   Revised:    MAY 31 2016
%   Revised by: ANDREA FELLING
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

FrameStartTimes=[];
FrameEndTimes=[];

    for i = 2:size(StrobeData,1)-1
        % if current data point is high, and next data point is low, the strobe
        % just dropped, indicating the start of a frame
        if StrobeData{i,'Strobe'}-StrobeData{i+1,'Strobe'}==1
            FrameStartTimes = [FrameStartTimes; (StrobeData{i,'Time_s'}+StrobeData{i+1,'Time_s'})
                /2];
        end
        % if current data point is high, and previous data point is low, the
        % strobe just rose, indicating the end of a frame
        if StrobeData{i,'Strobe'}-StrobeData{i-1,'Strobe'}==1
            FrameEndTimes = [FrameEndTimes; (StrobeData{i,'Time_s'}+StrobeData{i-1,'Time_s'})/2];
        end
    end

%% Concatonate into output variable
nFrames = min([size(RecordingData{:,'Frame'},1),size(FrameStartTimes,1),size(FrameEndTimes,1)]);
FrameStartTimes = FrameStartTimes(1:nFrames,:);
FrameEndTimes = FrameEndTimes(1:nFrames,:);
FrameData = [FrameStartTimes,FrameEndTimes];
if size(FrameData,1)<size(RecordingData{:,'Frame'},1)
    LinesMissing = NaN(size(RecordingData{:,'Frame'},1)-size(FrameData,1),2);
    FrameData = [FrameData;LinesMissing];
end
```