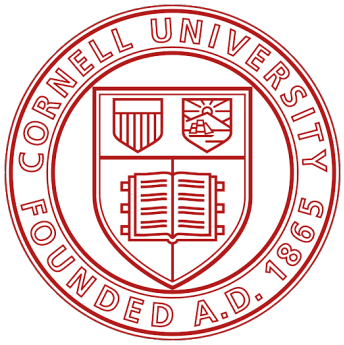

Cornell University
Cornell High Energy
Synchrotron Source
Ithaca, NY

Fort Lewis College
Department of
Physics and Engineering
Durango, CO

Digital Image Correlation @ CHESS (Boundary System Integration)

Authors:
Tyrone Bracker-Yazzie

Advisors:
Dalton Shadle

August 3rd
Summer 2020

Abstract

At CHESS we study the behavior of deformation in objects. Through the use of Digital Image Correlation (DIC) and x-ray measurements. The current version of the MATLAB DIC code suffers from "large deformation correlation loss"; it is our task to improve the correlation of our experiments along with giving the user more control over the code. Testing different options, we've proven what doesn't work to figure out how we can move in the right direction. The final product named "Boundary System" combines the initial incremental system by setting reference points where the stress and strain curve plot needs it the most.

1 Introduction

Digital Image Correlation (DIC) is a technique that measures the deformation of solid objects [1]. At CHESS the process of studying deformation uses both macroscopic (DIC) and microscopic (x-ray) techniques. We use DIC to measure and control the deformation during x-ray experiments at the materials beamlines. We have a MATLAB code dedicated to processing DIC data and plotting stress and strain in real-time. During a cyclic loading x-ray experiment at the Forming and Shaping Technology (FAST) Beamline at CHESS, a user employed the MATLAB code to monitor macroscopic strain in a sample and control deformation during their experiment. Initially, the user captured DIC strain measurements at small, incremental steps during loading. After several cycles, the user exchanged small, incremental DIC measurements for large, end-to-end DIC measurements. The change in procedure led to the DIC code improperly calculating macroscopic strain and plotting a stress-strain curve with some outliers (Fig. 1). These outliers are due to "large deformation correlation loss" which is when the code is trying to compare speckle patterns of a current image and a reference image that are too far apart. These events happened at the two extremes of the stress-strain curve (compression and tension) as they were the farthest away from the undeformed reference point. Our task was to reduce the amount of large deformation correlation loss. To begin we started with three potential solutions; the first solution was to compare each update to the reference data, the second solution was to set control-points at the extremes of the stress-strain curve, and the third solution was to create controllable correlation region axises. Our team implemented and managed to test the first two ideas.

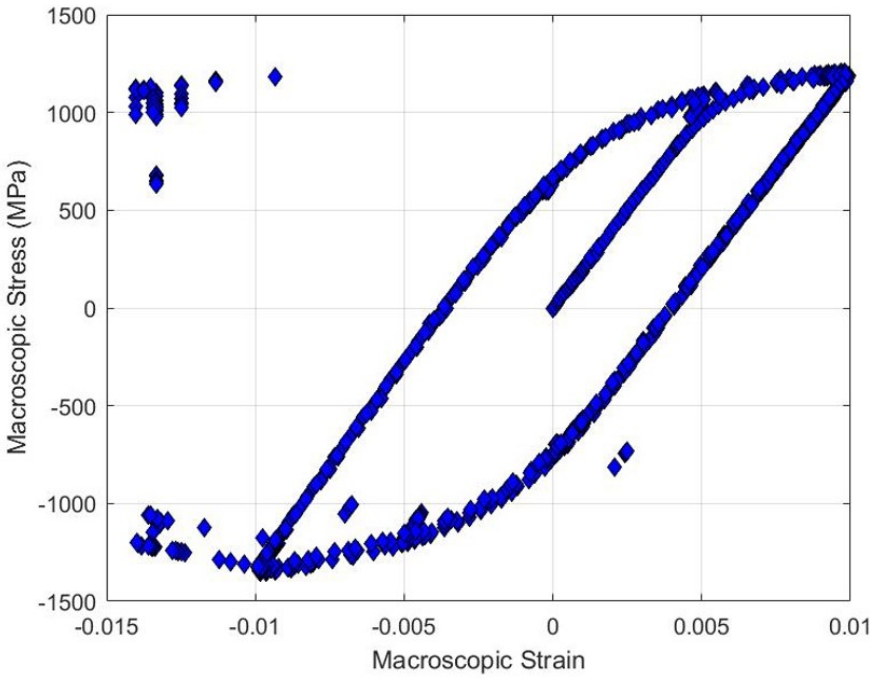

Figure 1: The figure depicts the stress-strain curve calculated by the MATLAB DIC code from DIC images captured during the experiment described above. At the top-left of the plot are the outliers due to large deformation corrrelation loss.

2 Background

`DICProcessingWCont.m` was the main script for running all other external DIC functions in MATLAB. One of the defining attributes for this code was its ability to process each DIC image and provide stress and strain values to plot.

During each deformation experiment, a dic.par file is generated that contains hardware info on the DIC measurements. `ProcessDicParFile.m` extracted image names and force values from the dic.par file for the program to process.

`SelectFirstImage.m` as the name implies selects the first image of the dic.par file. This function extracts the reference image and data.

Users of the experiment needed a way to visualize where the correlation regions and areas of interest on their objects were located, `GenerateGrid.m` was the answer to that. This allows the user to define an "area of interest" on an image. All other functions will focus on this area after it is created.

`ProcessCorrelations.m` defines the processing mode for correlation. This function can be thought of as the *gatekeeper* for `CPCorr.m` as the input values for `CPCorr.m` are checked and then sent for correlation.

The main correlation code `CPCorr.m` used a cross-correlation method to correlate two images for calculating displacement. `CPCorr.m` used 'fixed' and 'moving' points to create a fixed correlation region for the sub-region to move around inside [2]. The sub-region follows a speckle pattern while the object is being deformed to calculate displacements. Once the system updates with a new image, a 'current' image will update the sub-region to follow the speckle pattern it's been assigned.

`FitStrain.m` fits a polynomial of degree 1 to the coordinate and displacement values to calculate strain.

`ContinuousUpdate2020.m` continuously reads in DIC data from a dic.par file that is updated from the load-frame hardware in the lab. This function also housed most of the GUI for the boundary system. Since I wasn't able to attend Cornell physically, an external Python script was created and to help simulate what a user at a beamline would see as they were conducting a DIC experiment.

3 Methods

3.1 First Possible Solution

For the first possible solution, we explored improving correlation by altering `ProcessCorrelations.m`. ValidX/Y are the current positions of the grid pattern from the current image. ValidRefX/Y are the reference positions of the grid pattern from the first image. ValidX/Y were updated after each image as an estimate for the next grid pattern, however this can lead to bad estimates if the deformation is too large. Replacing the current values ValidX/Y in `ProcessCorrelations.m` with the reference values ValidRefX/Y would give us a static, central estimate for the grid pattern. Below in Listings 1 and 2 is a snippet of the code from `DICProcessingWCont.m` that exists at the beginning of the main For Loop. Line 9 is where the significant changes were applied.

Listing 1: Code Snippet Before

```
1 for i=1:NumFiles
2     ValidImage = [Image.Leader num2str(sprintf('%06d',File(FileIndexes
          (i),1))) Image.Extension];
3     ValidImageFullName = [FirstImage.Folder ValidImage];
4
5     if i==1
6         ValidX=ValidRefX;
7         ValidY=ValidRefY;
8     else
```

```
9           [ValidX,ValidY]=ProcessCorrelations(FirstImage.FullName,
                ValidImageFullName,GridX,GridY,ValidX,ValidY,corrsize);
10      end
```

Listing 2: Code Snippet After

```
1   for i=1:NumFiles
2       ValidImage = [Image.Leader num2str(sprintf('%06d',File(FileIndexes
            (i),1))) Image.Extension];
3       ValidImageFullName = [FirstImage.Folder ValidImage];
4
5       if i==1
6           ValidX=ValidRefX;
7           ValidY=ValidRefY;
8       else
9           [ValidX,ValidY]=ProcessCorrelations(FirstImage.FullName,
                ValidImageFullName,GridX,GridY,ValidRefX,ValidRefY,
                corrsize);
10      end
```

3.2 Second Possible Solution

For the second possible solution, we created a "boundary system" by setting new reference points at the two extremes of the stress-strain curve.

1. To test the feasibility of this solution, the first version created minimum and maximum values by inputting two boundary stress values for the current stress value to cross. The maximum and minimum reference values are the calculated grid locations at the tension and compression tips. For example: the boundary values are 700 MPa and -700 MPa, if the current stress value is above 700 MPa we would create and use a maximum reference value; if it is below -700 MPa we would create and use a minimum reference value; anywhere in between 700 MPa and -700 MPa is where we would use the default reference values (ValidRefX/Y).

2. To give the user more influence on what they wanted the maximum and minimum values to be, now named tension and compression tip(s), a plot GUI was implemented. Two buttons for compression and tension, displayed information about their current ValidX/Y, screw position, stress, and strain values. The values saved from the compression and tension buttons are then used for the *upper* and *lower* limits of the boundary system. The upper limit is the midpoint between the reference and tension tip screw positions. The lower limit is the midpoint between the reference and compression tip screw positions. Based on where the current screw position falls in the boundary system determines the estimate grid pattern we use in correlation. TensionX/Y and CompressionX/Y are the estimate grid patterns that're saved and used *based on* which section of the boundary system the user is currently in. These sections and variables can be seen below in Listing 3.

Listing 3: Boundary System Foundation Code

```
ScrewPos = File(end,3);
if ScrewPos > upperLimit
    [ValidX,ValidY]=ProcessCorrelations(FirstImageFullName,
        NewestImage,GridX,GridY,tensionX,tensionY,corrsize);
elseif ScrewPos < lowerLimit
    [ValidX,ValidY]=ProcessCorrelations(FirstImageFullName,
        NewestImage,GridX,GridY,compressX,compressY,corrsize);
else
    [ValidX,ValidY]=ProcessCorrelations(FirstImageFullName,
        NewestImage,GridX,GridY,ValidRefX,ValidRefY,corrsize);
end
```

4 Results

4.1 Using only reference data

This result can be seen as a visualization of our initial problem, large deformation correlation loss, although much greater (Fig. 2). The plot told us two things:

1. That using just reference data does not improve correlation loss. The reference data lies at the very center of the stress-strain curve, which is why the compression and tension areas of the curve suffered from high correlation loss.

2. Due to the lack of correlation at the two extremes, we decided create control points at the two extremes of the stress-strain curve.

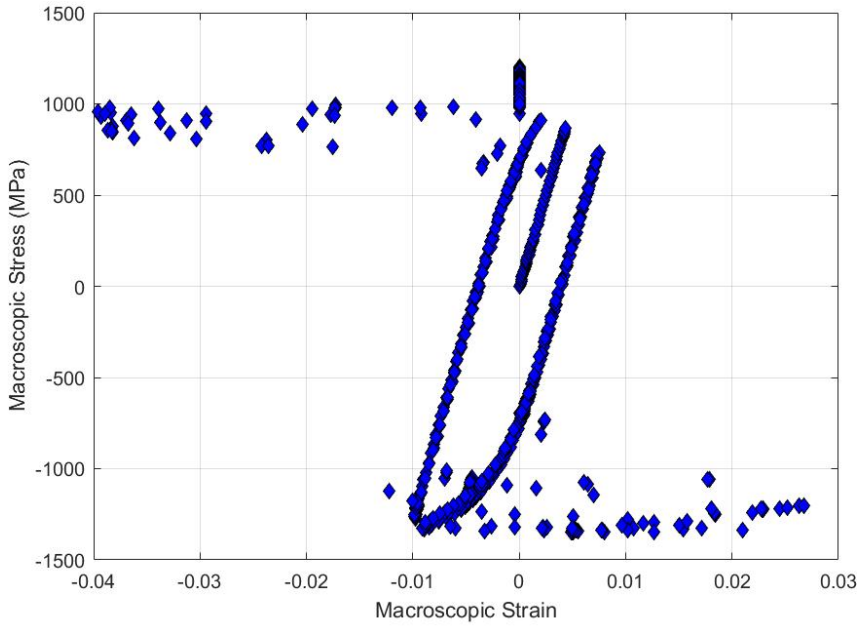

Figure 2: The figure above depicts the stress-strain curve calculated using only the reference grid pattern as an estimate for correlation processing. The point at the top-left and bottom-right are again due to large deformation correlation loss.

4.2 Setting reference points AKA Boundary System

With this boundary system that we've set up, the overall correlation has improved vastly. With a new user-friendly GUI implemented with pieces of information, the user will have a more visual plot window and the final product can be seen below (Fig. 3).

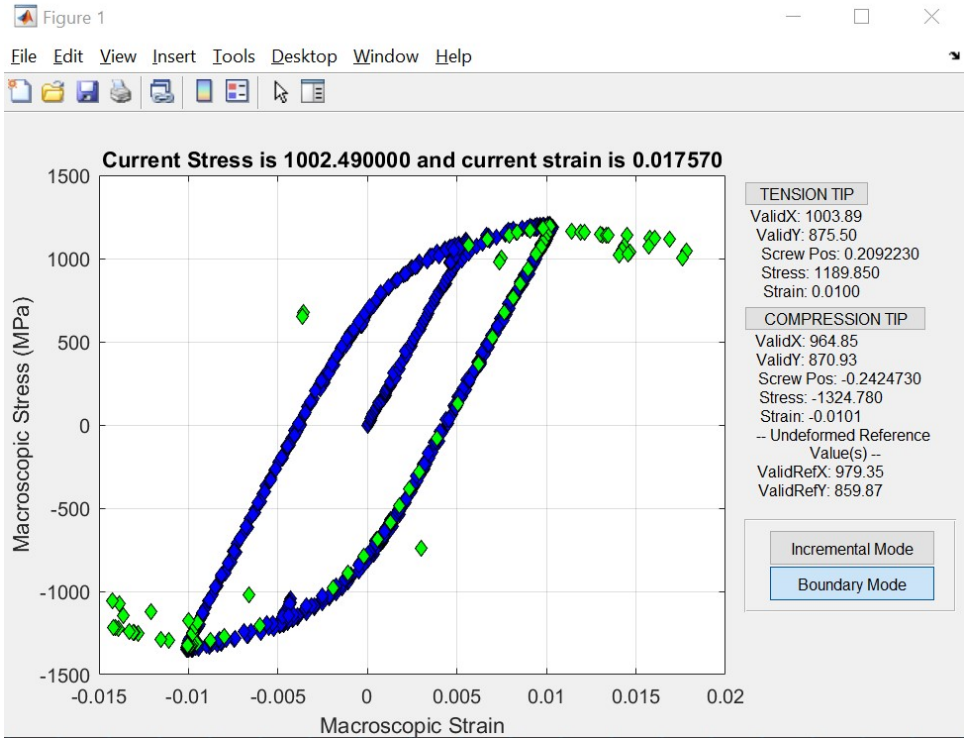

Figure 3: The user now has the ability to set reference points at the tension and compression tips as well as switch between the incremental system and boundary system. The green values are the new estimates after switching from the incremental system to the boundary system.

5 Future directions and Conclusions

The third possible solution that was not tested yet was the ability to control the correlation region's dimensions. Currently the shape of the correlation region is always square. The object deforms via stretching and on compressing, so if the correlation area could also mimic the deformity shape, we may be able to increase correlation. For example: the object is pulled and stretched, increasing in length and decreasing in height; the correlation area could also increase its length and decrease its height.

Digital Image Correlation can be applied to a variety of deformation studies. It's a key tool for macroscopic measurements, working with x-ray microscopic measurements. This can help us control deformation in certain objects and understand what is happening on either scales. The goal remains the same with this project, improve correlation.

Acknowledgements

I would like to thank the staff members of MSN-C, funded by AFRL Grant Number FA8650-19-2-5220 as well as CHEXS, funded by NSF Grant Number DMR-1829070. I would also like to thank Dr. Laurie Williams at Fort Lewis College for introducing me to the Cornell Summer Undergraduate Research in Science and Engineering (SUnRiSE) program for this opportunity.

References

1. LePage, W. *et al.* A practical guide to DIC. *Website.* `https://digitalimagecorrelation.org/` (July 2020).
2. Tune control point locations using cross-correlation. *Website.* `https://www.mathworks.com/help/images/ref/cpcorr.html` (July 2020).