Documentation for Matlab-based DIC code

E. M. C. Jones
University of Illinois

January 28, 2015

Version 3

1 Copyright

The author requests that users of this code notify the author of any modifications, improvements, or adaptations they make to this code, so that such modifications can be addressed and possibly incorporated in future versions of this code.

Contact information:
Elizabeth Jones
Beckman Graduate Fellow
NSF Graduate Research Fellow
Ph.D. Candidate in Theoretical and Applied Mechanics
University of Illinois at Urbana-Champaign
ElizabethMCJones@gmail.com
`www.linkedin.com/in/elizabethmcjones/`

2 Author Credit

This digital image correlation code is based on code written by Christoph Eberl, *et. al.* at the John Hopkins University. The original code and documentation can be found on Matlab's File Exchange: `http://www.mathworks.com/matlabcentral/fileexchange/12413`. This document describes the current code, which has been heavily modified by the current author, Elizabeth Jones.

3 Updates

3.1 Version 3

Version 2 of this code contained an error involving smoothing the displacements after deleting some grid points using the *delete_data_GUI*. Specifically, the following error would arise if two or more complete rows of grid points were deleted from the bottom of the region of interest:

Error using smooth_moving_average_V2>smooth_moving_average_loop (line 187)
Index exceeds matrix dimensions.

This error was related to a typographical error in the function *pad_disp_setup*. In Version 2, line 66 erroneously read:

Btemp = find(ind_col == B);

In version 3, this line was corrected to read:

Btemp = find(ind_row == B);

3.2 Version 2

Version 2 of this code contains the following updates:

3.2.1 Matlab Version and OS

Version 2 of the code has been tested and is compatible with Matlab versions R2012b and R2014b running on a PC with 64-bit Windows 7 Professional.

3.2.2 Correlation of the Images

- To more accurately reflect the algorithm used in the code and to be consistent with other DIC codes, the subset size was adjusted to accept only odd values . Previously, the code accepted only even values of the subset, but the actual subset cropped from the image around each control point was one pixel larger than the subset size entered in the *correlate_images_GUI*. That is, if a subset size of 20 was entered in the GUI, the code actually cropped a subset of pixels around the control point that was 21 x 21 pixels. Now, the cropped subset is exactly the size requested in the GUI.
- Fixed a bug concerning correlating images using the preceding image as a reference image and using initial guesses from the correlation results of reduced images. Previously, initial guesses generated by the correlation of reduced images were calculated with respect to the first image only. Now, they are calculated with respect to the correct reference image, either the first image or the preceding image, depending on the correlation mode of the full images.
- Added the option to control the threshold of the correlation coefficients for both reduced and full correlations.
- Added the option to control the search zone for both the reduced and full correlations.
- The values of the correlation coefficients are saved as the variable *corr_coeff_reduced* for the correlation of the reduced images or *corr_coeff_full* for the correlation of the full images. These variables are saved in the .mat files *disp_reduced_data* and *valid_data* respectively.

3.2.3 Computation of the Data

Version 1 of the code contained an error in the algorithm used to smooth the displacements before calculating the strains. In certain cases, when displacements were large enough that the control points near the edges of the images moved out of the field of view of the images, the smoothing algorithm did not properly pad the displacements near the edges of the region of interest (ROI). This resulted in inaccurate strain calculations near the edge of the ROI. Version 2 of the code implements a new built-in interpolation/extrapolation function, *scatteredInterpolant*, to properly pad the displacements even in cases of large displacements. Note that the new smoothing algorithm requires a longer computation time; therefore, parallel computing is recommended for large image sets.

The function *scatteredInterpolant* is not available in older versions of Matlab. For people using older version of Matlab, the smoothing algorithm used in Version 1 of this code is utilized. As long as displacements are small enough that the control points do not leave the field of view, the two smoothing algorithms should give similar results.

The use of *scatteredInterpolant* also provides more accurate strain calculations near the borders of regions of deleted data (i.e. near the edges of a crack or a hole).

Finally, the user now has more control over how the smoothing algorithm handles regions of uncorrelated data. An additional control has been added to the *compute_data_GUI* that allows the user to choose the maximum size of a group of contiguous uncorrelated points that the code will smooth over. If there is a group of neighboring uncorrelated points larger than this maximum, the code will not extrapolate data over that region.

3.2.4 Visualization of the Data

- Added the option to visualize either the raw displacements or the smoothed displacements from the same data set.
- Added the option to visualize the correlation coefficients so that the user can evaluate the confidence in the correlation results.
- Added the option to spatially average the data over a rectangular region that is a certain fraction of the region of interest (ROI) and is centered at the image center.
- Fixed a bug concerning plotting a contour plot of the reduced image correlation over a background image. Previously, the x- and y-coordinates used for the patch contour plot were in terms of the full-size image scale but the image itself was scaled by the "reduction" factor used for the reduced correlation. Now, the images and the results from the reduced correlation are scaled back to the scale of the full-sized images.

3.2.5 Evaluation of Accuracy and Precision of Code

The accuracy and precision of the code was evaluated using the test images from the DIC Challenge sponsored by the Society of Experimental Mechanics (SEM). A complete evaluation of the code is provided i the accompanying PDF document "DIC Challenge - Summary of Results."

4 Introduction

This is a Matlab-based code that performs digital image correlation. This document assumes that the reader has a basic knowledge of Matlab and of the digital image correlation process. The purpose of this document is to describe the code so that the reader can use the code to obtain displacements and strains from their own experiments and modify the code as necessary.

5 Configuration

5.1 Version and OS

Version 1 of the code was written in Matlab version R2012b, on a PC running 64-bit Windows 7 Professional, and was tested on both a PC and a Mac running Matlab version R2011a. Version 2 of the code was tested with Matlab version R2012b and R2014b, both on a PC running 64-bit Windows 7 Professional.

5.2 Toolboxes

Three Matlab Toolboxes are required or suggested:

1. *Image Processing Toolbox* is used extensively throughout this code and is therefore required.
2. *Parallel Computing Toolbox* is required if the user wishes to process multiple images concurrently, which greatly reduces processing time. However, the user may choose to process the images in serial, thereby avoiding the need for the Parallel Computing Toolbox.
3. *Statistics Toolbox* is used in the smoothing algorithm (*compute_data_GUI* → *smooth_disp* → *smooth_moving_average* → *normal_distribution* → *normcdf*).

5.3 Additional Code from FileExchange

The code makes use of two sets of code downloaded from Matlab Central File Exchange. For simplicity, these codes are contained in the Matlab DIC files in this FileExchange. **There is no need to download these files separately.**

1. *Parfor Progress Bar:* This progress bar allows the user to monitor the progress the code is making while executing "parfor" loops during parallel computing. For compatibility with Matlab version R2014b, version 2 of this DIC code contains V0.2.8 of the parfor progress bar code. `http://www.mathworks.com/matlabcentral/fileexchange/35609`
2. *Freeze Colors:* This code is used in *visualize_data_GUI* when plotting semi-transparent DIC data over background images. `http://www.mathworks.com/matlabcentral/fileexchange/7943-freezecolors-unfreezecolors`

5.4 Configuration Steps

There is minimal configuration required to set up this DIC code:

1. Place the DIC Matlab files in one folder, and add the folder to the Matlab search path, directly below the default folder. This needs to be done only once, when you first install the code.
   - In Matlab 2012b, go to the "Home tab". In the "Environment" box, click on "Set Path". Click on "Add Folder...", and navigate to the folder containing the DIC Matlab files. Move this folder directly below the default folder (usually "...\Documents\MATLAB").
   - In earlier versions of Matlab, "Set Path" is under "File".
2. Change the Image Processing Toolbox preferences to display axes on images when using "imshow."
   - In Matlab 2012b and 2014b, type "iptprefs" in the command window. Check the box for "IMSHOW Display - Axes visible."
   - Alternatively, type *iptsetpref('ImshowAxesVisible','on')* in the Command Window. You only need to do this once. The preference will be remembered.
3. Check to see if the message catalog "menu.xml" exists in the following location (or equivalent location): C:\Program Files\MATLAB\R2012b\resources\MATLAB\en\uistring\menu.xml. If the folder "uistring" does not exist, create a folder called "uistring" within the "en" folder. Then place the "menu.xml" file, included in the DIC files in this FileExchange, into the "uistring" folder. If the folder "uistring" exists and contains the "menu.xml" file, then you do not need to do anything.
4. Some Mac users may encounter an error, similar to the one found below, when running the code in parallel mode:

   >> matlabpool open

   Error using matlabpool (line 134)

   Java exception occurred:

   java.lang.NullPointerException

   at java.util.logging.Logger.demandLogger(Logger.java:286)

at java.util.logging.Logger.getLogger(Logger.java:321)

If you experience this issue, please see the bug report 919688 on the MathWorks website: `http://www.mathworks.com/support/bugreports/919688`. Alternatively, you can run the code in serial mode.

6 Code Structure

6.1 Main GUIs

There are four main components to this code, each with its own GUI. To run these GUIs, simply type the name of the GUI in the Matlab command window and press enter. No input arguments are needed. A more detailed description of the different GUIs is presented in Section 8.

1. *image_setup_GUI*: Prepares images
2. *correlate_images_GUI*: Performs the image correlation and outputs displacements
3. *compute_data_GUI*: Smooths displacements, and interpolates displacements and calculates strains using finite element shape functions
4. *visualize_data_GUI*: Displays displacements and strains in a variety of formats

Additionally, there are two supplementary GUIs that can be useful under certain circumstances, but are not required:

1. *delete_data_GUI*: Deletes data that did not correlate well
2. *movie_GUI*: Combines contour plots of correlated data from all images into a time lapsed movie

6.2 File Format

There are two main types of files generated by this code, "setup" files and "data" files. The "setup" files are named "XX_setup", and contain information about the correlation, such as the subset size used, the grid spacing, the number of images correlated, etc. These are used by the code, but are also available for your reference. To view the information contained in a "setup" file, simply load the data into Matlab by typing "load XX_setup" in the Command Window and then view the data by typing "XX_setup" in the Command Window. For example, to see the number of images correlated, type the commands as shown in Fig. 1.

The main data generated by the code is stored in ".mat" files saved in the working directory. To manipulate the data directly, first load the data into Matlab by typing "load XX" in the Command Window, where "XX" is the file name. Note that several variables appear in the workspace when a single file is loaded. Each variable can be manipulated individually.

6.3 Steps of a Typical Correlation

The steps involved to run a typical correlation are found below. An example correlation using the provided images is found in Section 7.

1. Set up the images to be correlated.
   a) Place all images to be processed in one folder.
   b) Set the working directory in Matlab to the folder containing the images.
   c) Run *image_setup_GUI* to prepare the images.
2. Determine if displacements are large enough to require an initial guess.
   a) Run *correlate_images_GUI*, and correlate full-sized images only using a sparse grid (step size of about 50-75 pixels for images about 1000x1000 pixels). Using a sparse grid reduces computation time.
   b) Run *visualize_data_GUI* and look at the contour plot of the displacement you anticipate to be larger.

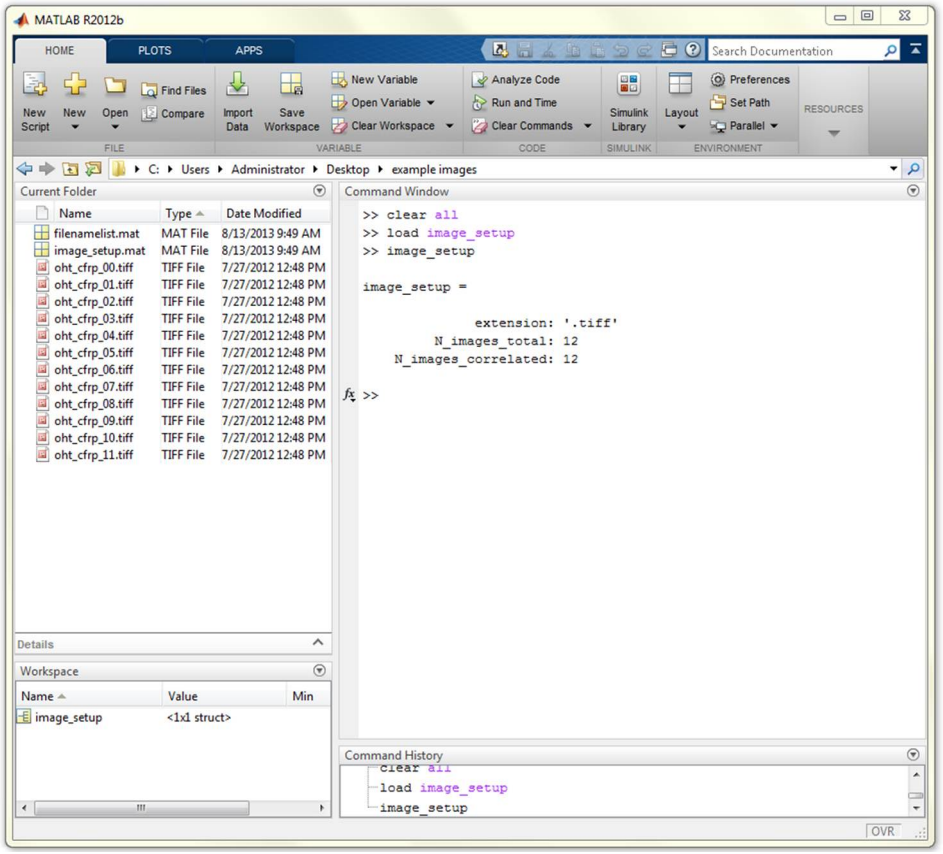


Figure 1: Example of loading a setup file, in this case *image_setup*, to see the parameters used for a correlation.

c) If the entire region of interest correlated well (there are no large areas of missing data where there are large displacements), then displacements are small enough that no initial guess is required, and the reduced-size images do not need to be correlated. Proceed to step 4.

d) If the images did not correlate well in regions where displacements are large, then you need to generate initial guesses for the displacements by correlating reduced-size images. Proceed to step 3.

3. If necessary, generate initial guesses for displacements by correlating reduced-size images.

   a) Run *correlate_images_GUI*, and correlate reduced-size images only. Use approximately 100-200 grid points.

   b) Run *visualize_data_GUI* and evaluate the results.

   c) If the correlation of the reduced images did not capture the large displacements, adjust one or more of three parameters until satisfactory results are obtained:

      - Change the subset size.
      - Change the image reduction factor.
      - Iterate the correlation of the reduced images, using the results from the previous correlation of reduced images as initial guesses for the current correlation of reduced images.

4. Determine the optimal subset size for the correlation of the full-sized images[1].

   a) Run *correlate_images_GUI*, and correlate full-sized images only, using results from the correlation of the reduced-size images as initial guesses as necessary, and using a sparse grid (step size of about 50-75 pixels for images about 1000x1000 pixels).

   b) Run *visualize_data_GUI* and evaluate the results.

[1] A fundamental assumption of this DIC code is that each subset undergoes only rigid translation in two directions; that is, that there is no deformation (or rigid rotation) within a single subset. For this reason, it is desirable to make the subset as small as possible. However, one must have a sufficiently large subset in order to have enough information within the subset to have a good correlation. The optimal subset size is a balance between these two competing requirements.

c) If there are very few points that did not correlate, then run *correlate_images_GUI* again, using a smaller subset size.

d) Repeat this process until the subset size is too small to provide a good correlation. Note the smallest subset size that provided a good correlation, and use this in the final correlation.

5. Run final correlation.

a) Run *correlate_images_GUI*.

b) Correlate full-size images only.

c) Use results from the correlation of reduced-size images if necessary.

d) Use the optimal subset size found in the previous step.

e) Create a new, dense grid (step size of approximately 5-10 pixels for images about 1000x1000 pixels).

6. Delete regions of poorly-correlated data. (Optional)

a) Run *visualize_data_GUI*.

b) Determine if there are regions where data did not correlate well.

c) Run *delete_data_GUI*.

d) Select a single image to preview, and choose regions of data to delete. Data will be deleted for all the images.

7. Smooth displacements and calculate strains.

a) Run *compute_data_GUI*.

b) If desired, enter the scale of your images (must be determined separately).

c) Choose a smoothing kernel size and the number of smoothing passes.

d) Choose finite element used in the interpolation of displacements and subsequent strain calculations. The author recommends always using cubic elements.

e) Compute the deformed grid if desired.

f) Run *visualize_data_GUI* to view the smoothed displacements and strains. Line scans of displacements and of strains are particularly useful to judge the effectiveness of the smoothing.

g) Re-run *compute_data_GUI*, adjusting the smoothing parameters as necessary until smoothed displacements and strains are satisfactory.

8. Visualize results.

a) Run *visualize_data_GUI*.

b) Save plots as desired.

c) Run *movie_GUI*. Make a time-lapsed movie of the contour plot if desired.

7 Example Correlation

An example correlation is provided here that follows the steps outlined in Sec. 6.3. The images used in the example correlation are from the DIC Challenge, presented by the Society of Experimental Mechanics (SEM), image set 12. More information on the DIC Challenge can be found in Sec. 9 and at `www.sem.org/dic-challenge/`.

7.1 Image Setup

Set up the images to be correlated.

1. Place all the example images into one folder. In this example, the folder is called "example images" and is located on the Desktop.

2. Set the working directory in Matlab to the folder containing the images.

3. (Fig. 2) Run *image_setup_GUI* by typing "image_setup_GUI" into the Command Window and pressing Enter.

4. (Fig. 3) Choose the appropriate file extension for the images. For this example, choose ".tiff". Set "Image skip" to 1 in order to correlate all the images in the folder.

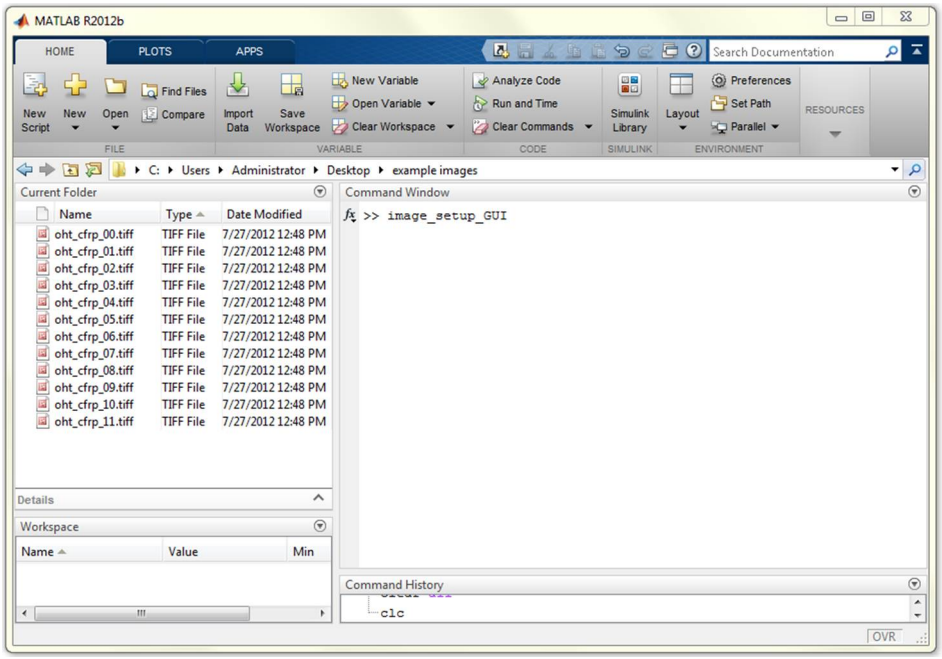


Figure 2: The working directory is set to the folder containing the example images. Run *image_setup_GUI* by typing "image_setup_GUI" into the Command Window and pressing Enter.

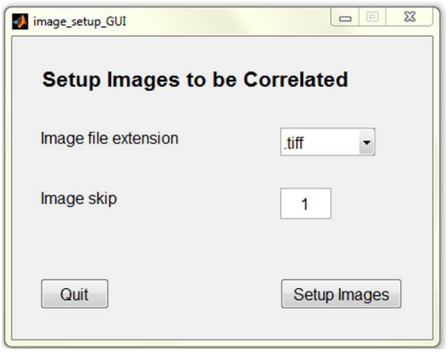


Figure 3: Options available for preparing the images.

7.2 Large Displacements

Determine if displacements are large enough to require initial guesses.

1. Correlate the full images using a sparse grid.

   a) (Fig. 4) Run *correlate_images_GUI* by typing "correlate_images_GUI" into the Command Window and pressing Enter.

   b) (Fig. 5) Choose to run in parallel or serial. This example uses parallel computing. If you choose to run in serial, choose "First Image" as the reference image. Correlate full images only, and define a new full grid. Use a subset of 21 pixels.

   c) (Fig. 6) If you choose to use parallel computing, the matlabpool must initialize. This happens only once every time you open Matlab.

   d) (Fig. 7) When prompted, select the reference image (first image) to open. When the image opens in Matlab, click on the top left corner and the bottom right corner to define the region of interest. Choose a step size of 50 pixels. When prompted, choose to "Keep this grid." The images will automatically be correlated, and the data from the correlation will be saved in the working directory.

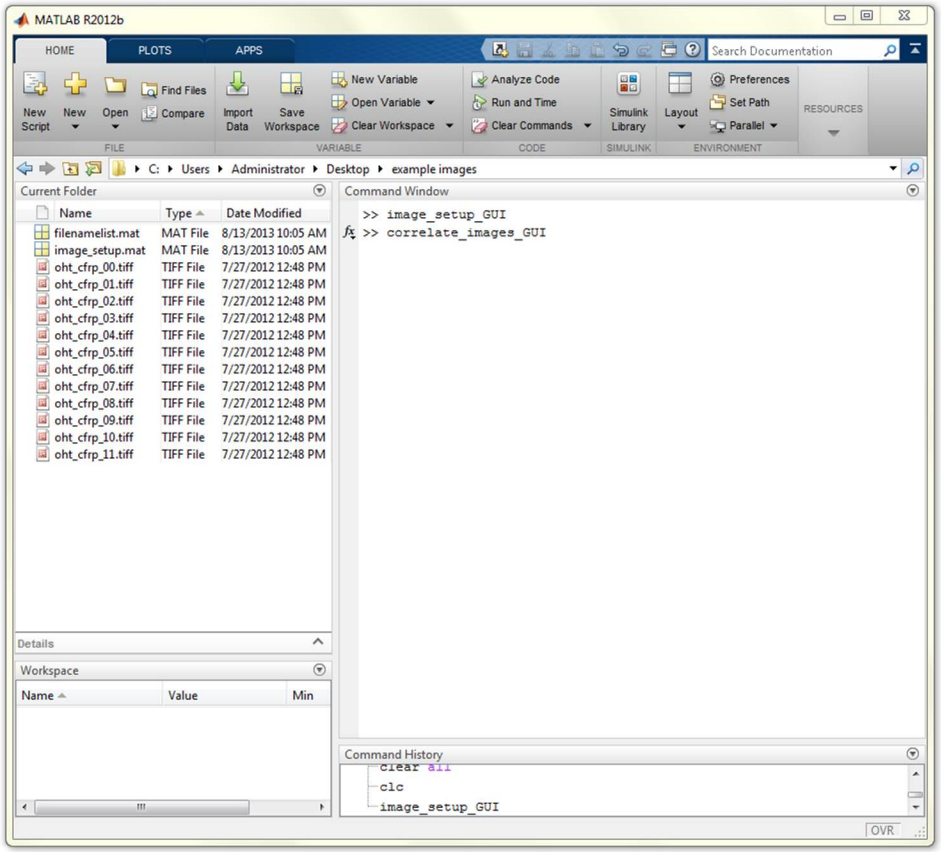


Figure 4: Run *correlate_images_GUI* by typing "correlate_images_GUI" into the Command Window and pressing Enter.

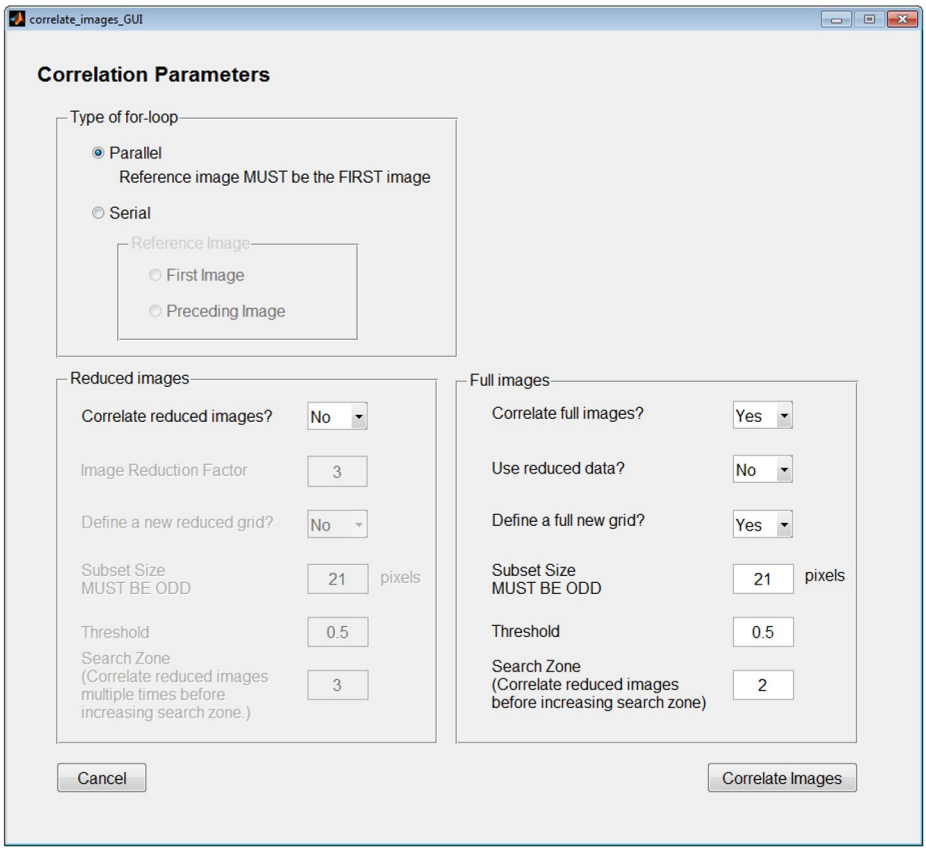


Figure 5: Correlation parameters for the correlation of full-sized images, used to determine if initial guesses are required to capture large displacements. If running in serial mode, choose the first image as the reference image.

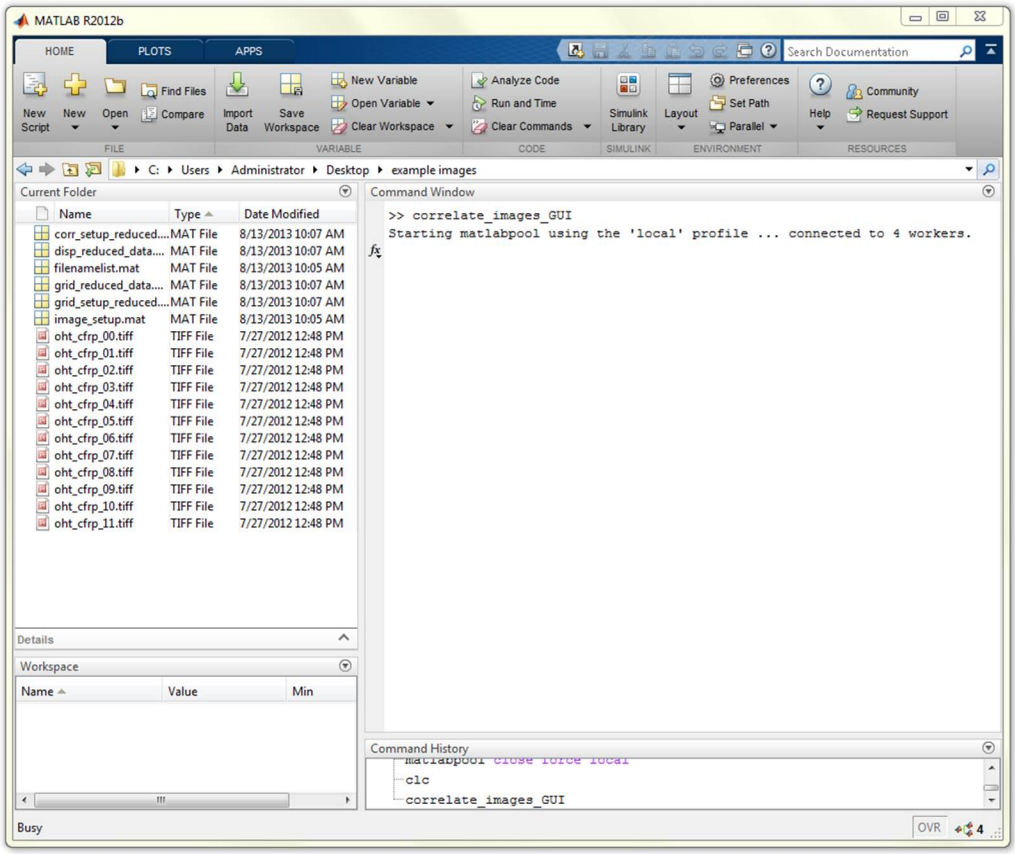


Figure 6: The first time parallel computing is used after Matlab is opened, the matlabpool must initialize.

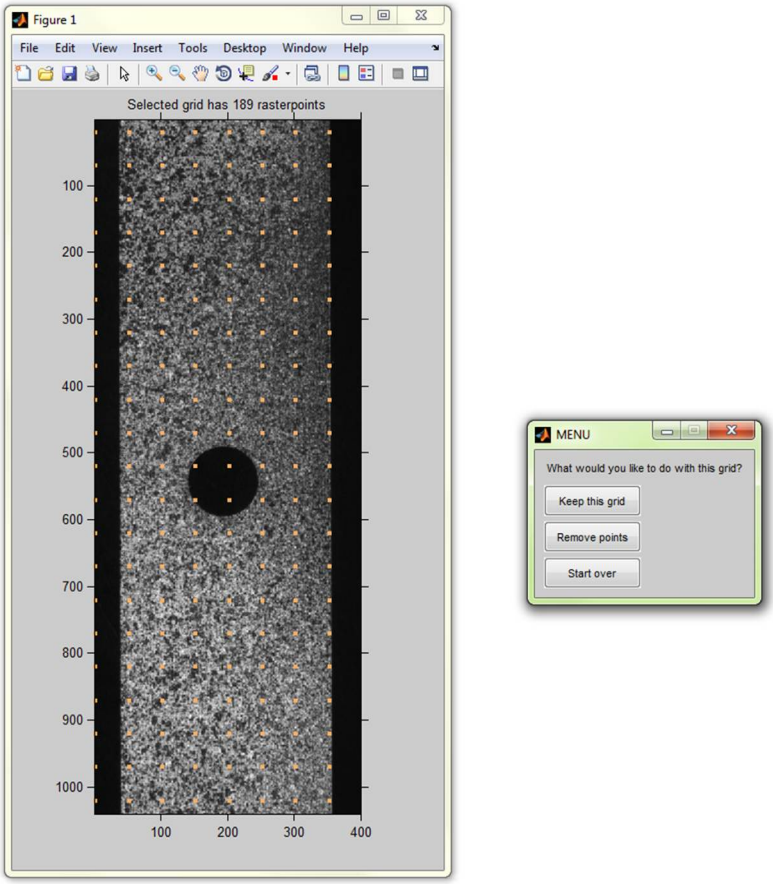


Figure 7: Sparse grid (step size of 50 pixels), defined on a full-size image.

2. Visualize the results to determine if the images correlated well.

   a) (Fig. 8) Run *visualize_data_GUI* by typing "visualize_data_GUI" in the Command Window and pressing Enter.

   b) (Fig. 9) Visualize the full data (data from the correlation of full-sized images). Click on "Filled Contour Plot". To view data from all of the images, leave "Image Skip" set to 1.

   c) (Fig. 10) Choose to view the vertical displacements (V), and use the same scale for all of the images. Ignore the "Plot over images?" box in order to plot the contours without a background image. Click on "View Contour Plot."

   d) (Fig. 11) The first image is a self-correlation, and should have nearly zero displacements. Images 2 though 8 correlated over the entire region of interest, while images 9 through 12 did not correlate near the top of the region of interest, where the displacements are large. Therefore, initial guesses for the displacements are required.

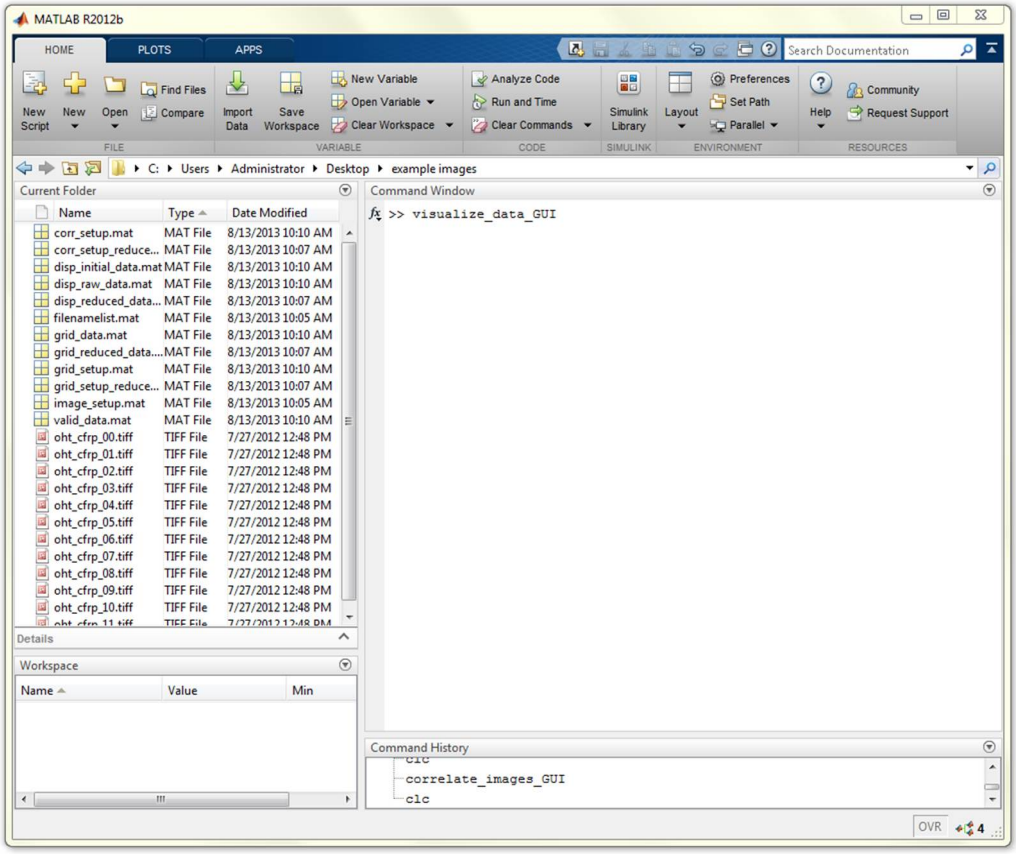


Figure 8: Run *visualize_data_GUI* by typing "visualize_data_GUI" into the Command Window and pressing Enter.

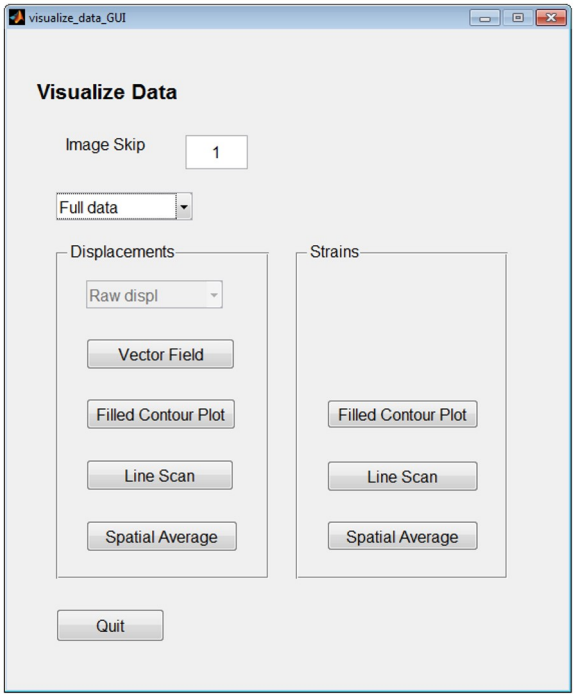


Figure 9: Options available for visualizing the results of the correlation of the full-size images. For this example, choose "Filled Contour Plot."

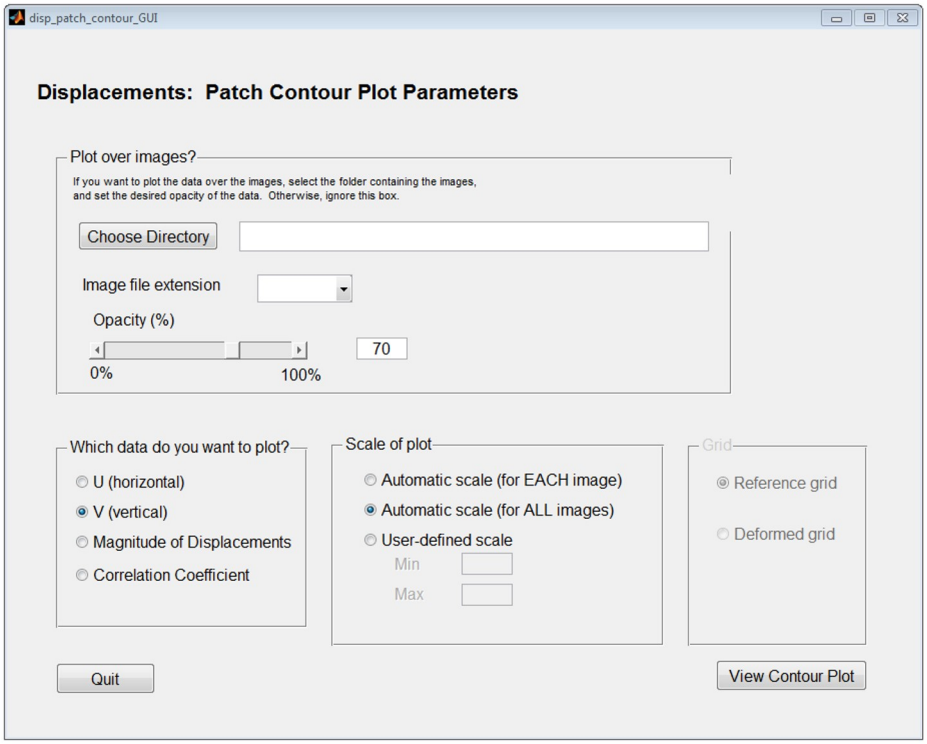
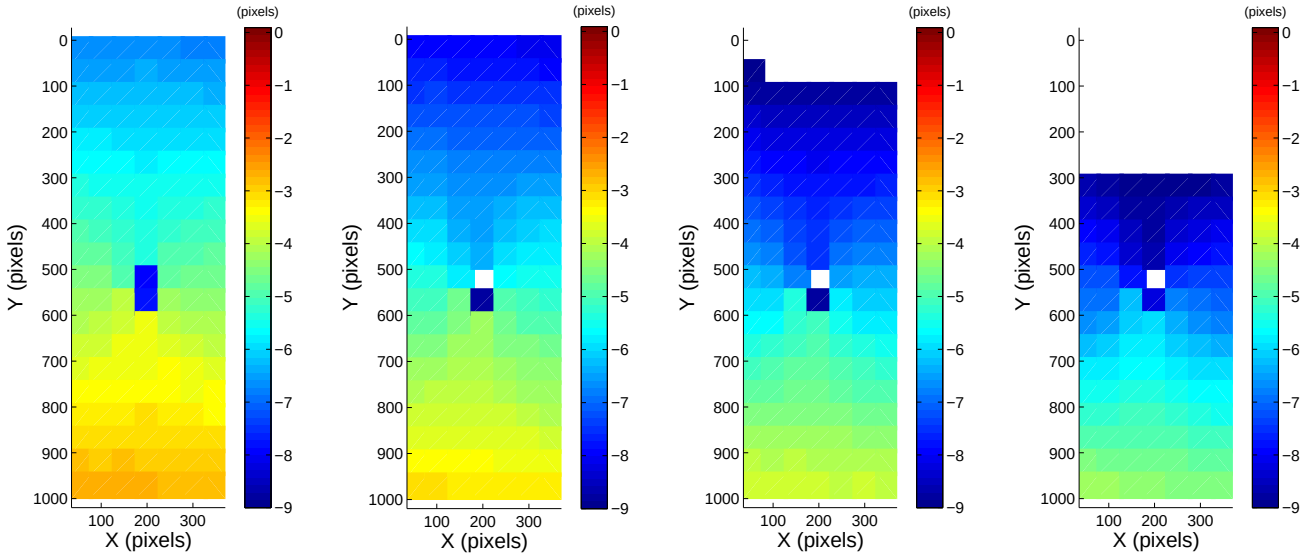


Figure 10: Options available for visualizing contour plots of displacements. For this example, choose the vertical displacements, V, and use the same scale for all images.



(a) Image 7 (b) Image 8 (c) Image 9 (d) Image 10

Figure 11: Contour plots of the vertical displacements, V, for images 7-10 for the first correlation on a sparse grid of the full-size images without using initial guesses. Images 2-8 correlated well, but images 9-12 have large regions of data missing near the top of the region of interest, where there are large displacements.

7.3 Generate Initial Guesses (Correlate Reduced Images)

Generate initial guesses for displacements by correlating reduced-size images.

1. Correlate the reduced-size images.

   a) Run *correlate_images_GUI*.

   b) (Fig. 12) Correlate the reduced-size images only. Set the correlation parameters as shown in Fig. 12.

   c) (Fig. 13) When prompted, select the reference image (first image) to open. Note that the image is now shown in the reduced size. Click on the top left and bottom right corners of the image to define the region of interest. Choose a step size of 20 pixels. Keep this grid. The reduced-size images will automatically be correlated.

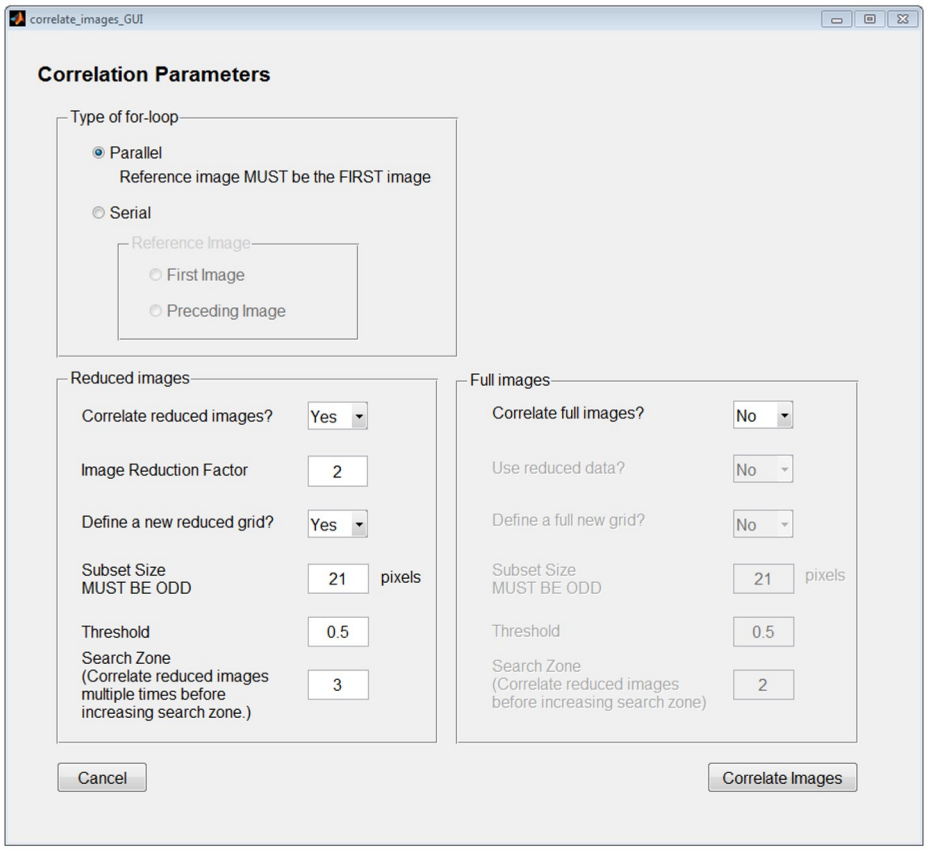


Figure 12: Correlation parameters for the correlation of reduced-size images, used to generate initial guesses for the correlation of full-sized images.

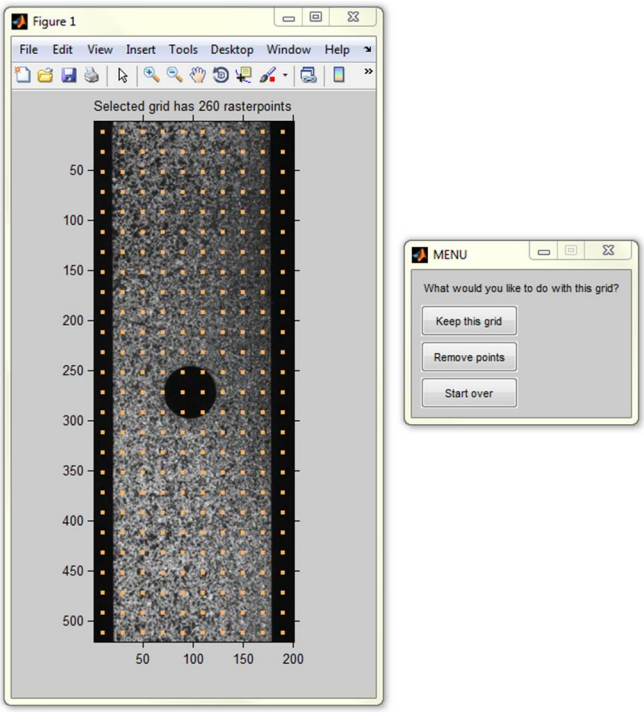


Figure 13: Grid used for the correlation of the reduced-size images. The step size was 20 pixels.

2. Visualize the results of the correlation of the reduced-size images to determine if the images correlated well.

   a) Run *visualize_data_GUI*.

   b) (Fig. 14) Visualize the reduced data (data from the correlation of the reduced-size images). Click on "Filled Contour Plot." To view data from all of the images, leave "Image Skip" set to 1.

   c) Choose to view the vertical displacements (V), and use the same scale for all of the images. Click on "View Contour Plot."

   d) (Fig. 16a) The large displacements for all images were captured effectively by the reduced-size images. These results will provide initial guesses for the final correlation of the full-sized images. There is no need to re-correlate the reduced-size images; future correlations of the full-sized images will use the data from this correlation, which is saved in the working directory, as initial guesses.

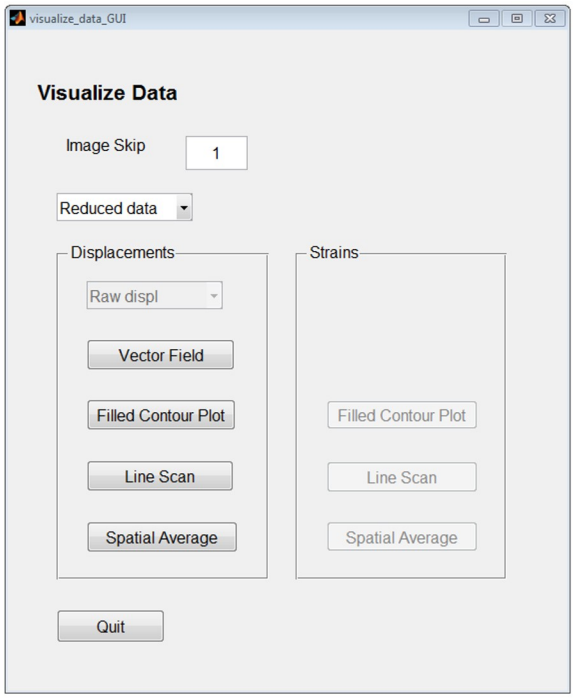


Figure 14: Options available for visualizing the results of the correlation of the reduced-size images. For this example, choose "Filled Contour Plot."

3. Correlate the full-sized images using the results from the correlation of the reduced-size images as initial guesses.

   a) Run *correlate_images_GUI*.

   b) (Fig. 15) Correlate the full-sized images only. Use the results of the correlation of the reduced-size images from the previous step as initial guesses. Use the same sparse grid that was defined in Sec. 7.2.

   c) Run *visualize_data_GUI*, go to "Full data", and look at the contour plot of the vertical displacements.

   d) (Fig. 16b) Verify that all regions of large displacements are captured using the initial guesses. Note that the very top row of data may be missing in images 11 and/or 12. This is due to the top portion of the sample moving out of the field of view of the camera. It is NOT a result of the displacements being too large.

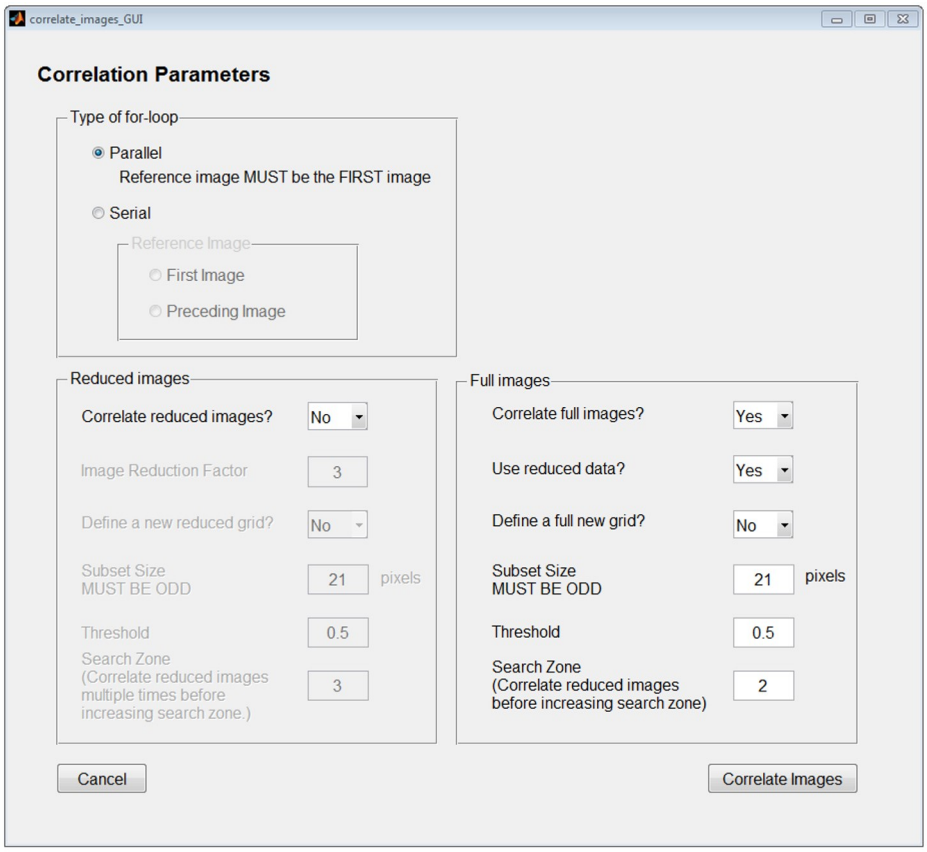
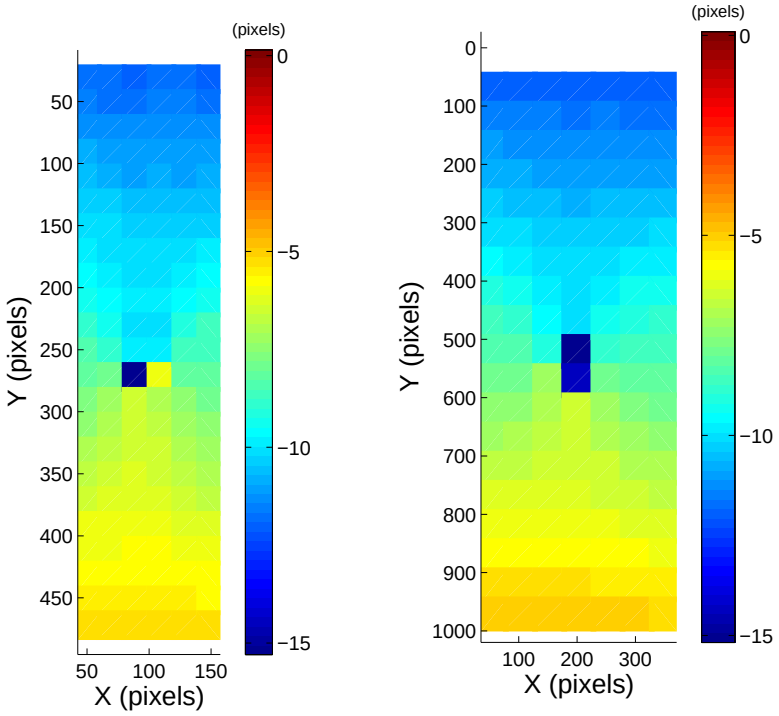


Figure 15: Correlation parameters for the correlation of full-size images, using the results of the correlation of the reduced-size images as initial guesses.



(a) Reduced correlation

(b) Full correlation, with initial guesses

Figure 16: Contour plot of the vertical displacements, V, for the last image (image 12) for (a) the reduced-size images and (b) the full-size images using initial guesses. All large displacements are captured well.

7.4 Optimal Subset Size

Determine the optimal subset size for the correlation of the full-sized images.

1. Correlate the full images, using initial guesses if necessary, using a smaller subset size.

   a) (Fig. 17) When using a subset size of 21 pixels in the previous step, all the subsets correlated well, suggesting that a smaller subset may be sufficient. Run *correlate_images_GUI*. Correlate full images only, using the reduced data, using the same sparse grid that was previously defined, but use a smaller subset of 11 pixels.

   b) Run *visualize_data_GUI*, go to "Full data", and look at the contour plot of the vertical displacements.

   c) (Fig. 18) All of the data points correlated well, indicating that a smaller subset could be used.

   d) (Fig. 18) Repeat the previous steps, using a subset size of 7 pixels, and then repeat the steps using a subset size of 5 pixels. The correlations for subset sizes of 21, 11, and 7 are all nearly equivalent. The correlation for subset size of 5, however, is missing several data points, indicating that the subset size of 5 contains insufficient data to provide robust correlation results. Therefore, the optimal subset size is 7 for this particular set of images.

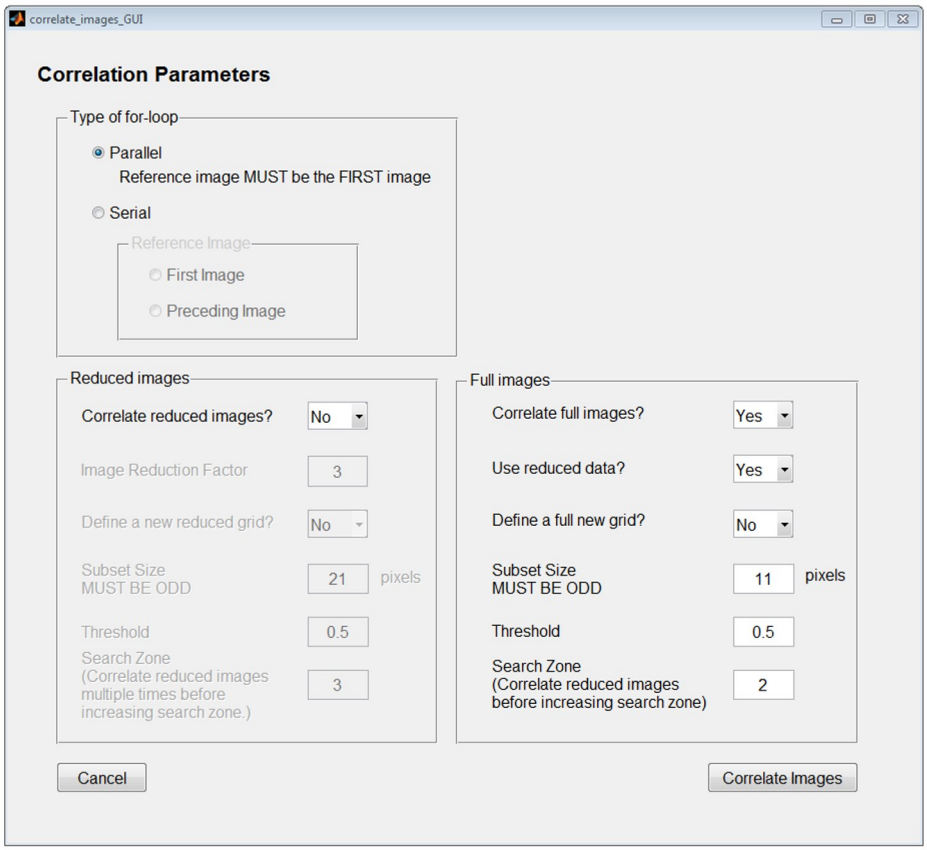
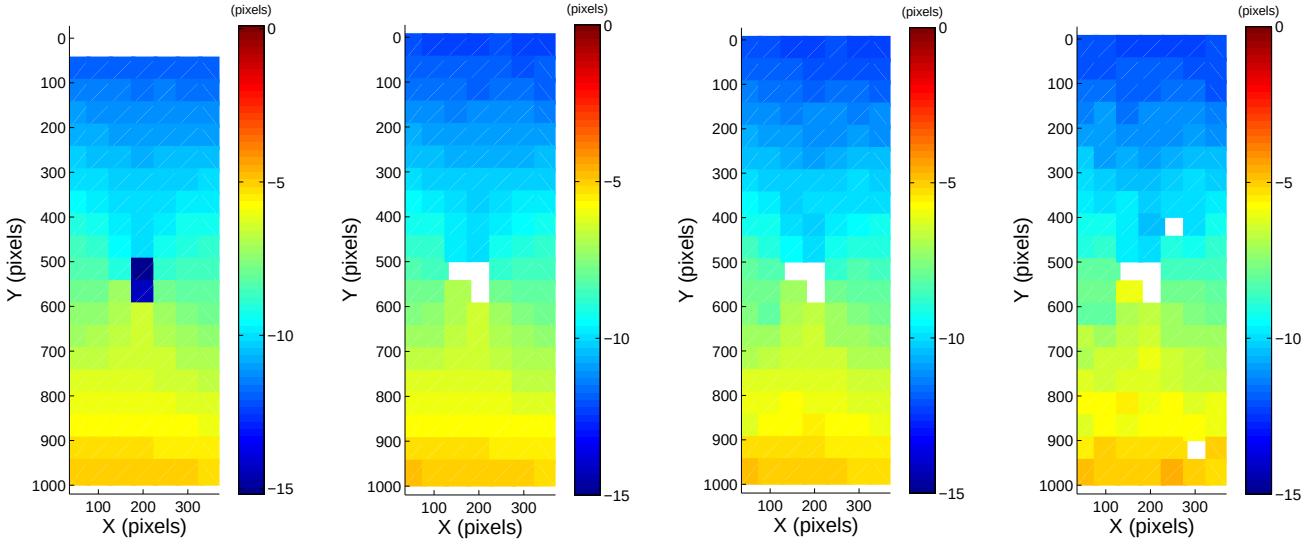


Figure 17: Correlation parameters for the correlation of full-size images, using a smaller subset size.



(a) Subset 21 (b) Subset 11 (c) Subset 7 (d) Subset 5

Figure 18: Contour plots for the vertical displacements for the last image (image 12) for four different correlations with varying subset sizes. Correlations with subsets larger than 7 pixels correlated well, while the correlation with a subset size of 5 pixels did not correlation well. Therefore, the optimal subset size for this set of example images is 7 pixels.

7.5 Final Correlation

Run the final correlation, using a dense grid.

1. Correlate the full images, defining a new, dense grid.

   a) (Fig. 19) Run *correlate_images_GUI*. Choose the parameters as shown in Fig. 19.

   b) (Fig. 20) When prompted, define a new grid with a step size of 5 pixels.

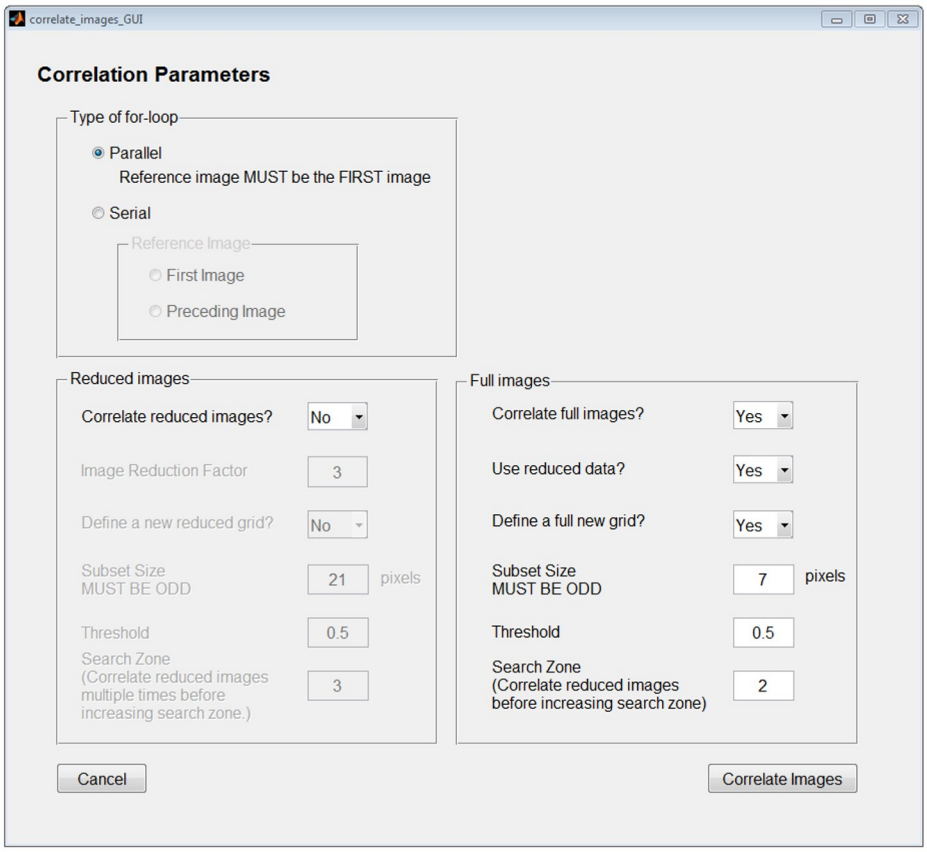


Figure 19: Correlation parameters for the correlation of full-size images, where a new grid will be defined.

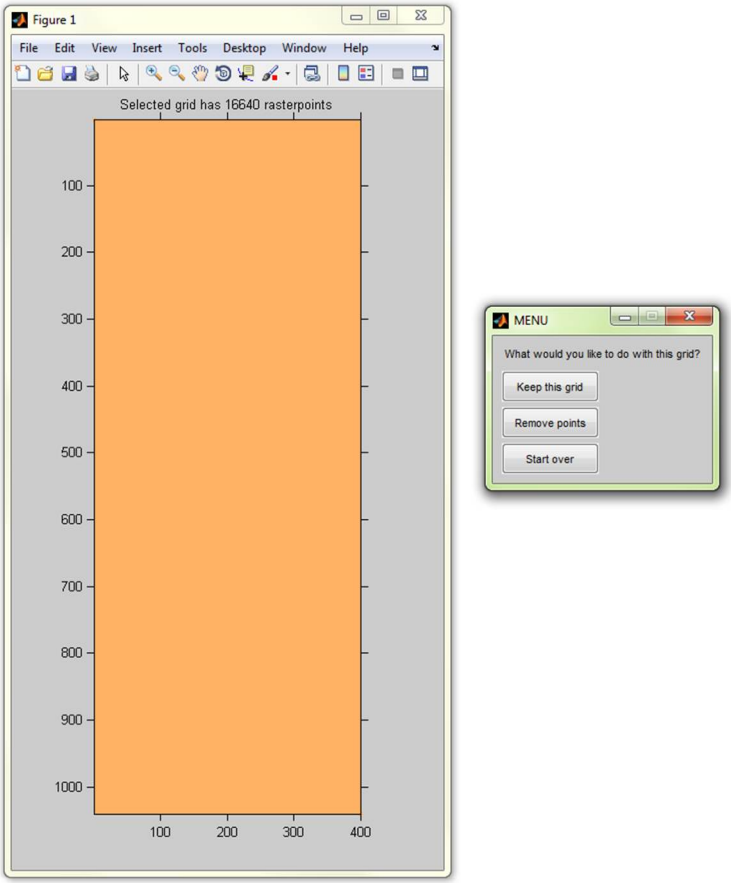


Figure 20: Dense grid (step size of 5 pixels) used for the final correlation of full-size images. Note that the grid is so dense, it appears as a solid rectangle.

2. Visualize the results of the correlation with a dense grid.

   a) (Fig. 21) Run *visualize_data_GUI*, and look at the contour plots of the vertical displacements.

   b) Notice that for the most part, the grid points that were not over the test specimen (i.e. at the left and right edges of the image and in the hole in the specimen center), did not correlate. However, there are some false correlations in these regions. These false correlations can affect the strain results if they remain in the data. Therefore, they should be deleted before smoothing displacements and calculating strains.

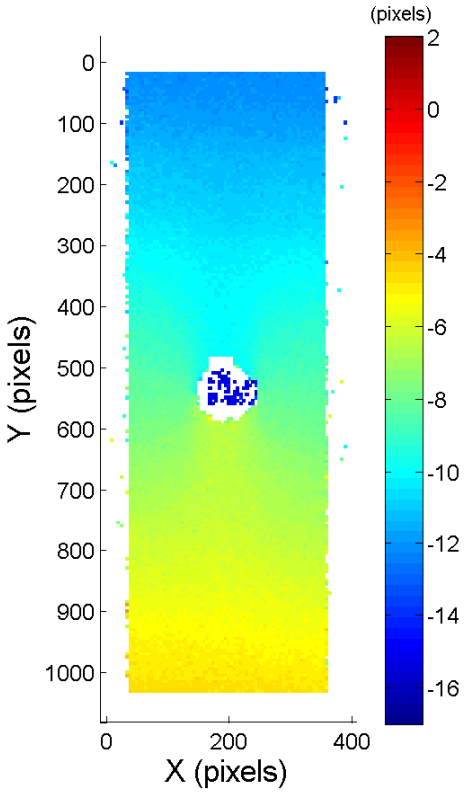


Figure 21: Contour plot of the vertical displacements, V, for the last image (image 12) after the final correlation of the full-size images using a dense grid.

7.6 Delete Poorly Correlated Data

Delete regions of poorly-correlated data.

1. Save a copy of the original data in a separate folder. This saves you from needing to re-do the correlation if you accidentally deletes too much data.

2. Delete poorly-correlated data.

   a) Run *delete_data_GUI* by typing "delete_data_GUI" into the Command Window and pressing Enter.

   b) (Fig. 22) Choose image 1 to preview. Choose "Grid Points" as the data representation. When prompted, select the reference image (first image) to open. This will plot the original grid over image 1.

   c) (Fig. 23a) Click on the top left and bottom right corners of the region you wish to delete. If necessary, choose "Yes" when prompted with "Would you like to select more points to delete?" and select more regions until all the grid points that are not over the specimen have been selected and temporarily deleted.

   d) Note: Once you select data, you cannot deselect it. However, data is not deleted until you click on "Delete Data." Therefore, if you accidentally selected data you don't want to delete, simply close the figure without clicking on "Delete Data." You can then, click on "Grid Points" again, and select more data.

   e) When you've selected all the desired data to delete, click on "Delete Data" to permanently delete the data.

   f) When you're finished using the *delete_data_GUI*, click on "Done / Quit."

3. Visualize correlation results with poorly-correlated data deleted.

   a) Run *visualize_data_GUI* and look at the contour plots of the vertical displacements.

   b) (Fig. 23b) There is no longer any falsely-correlated data at the edges of the image and in the center hole of the specimen.

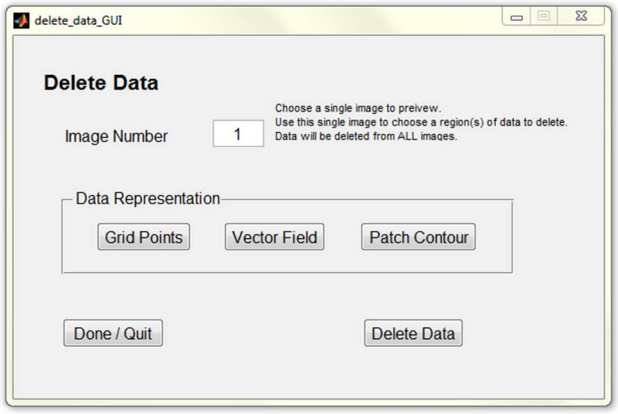
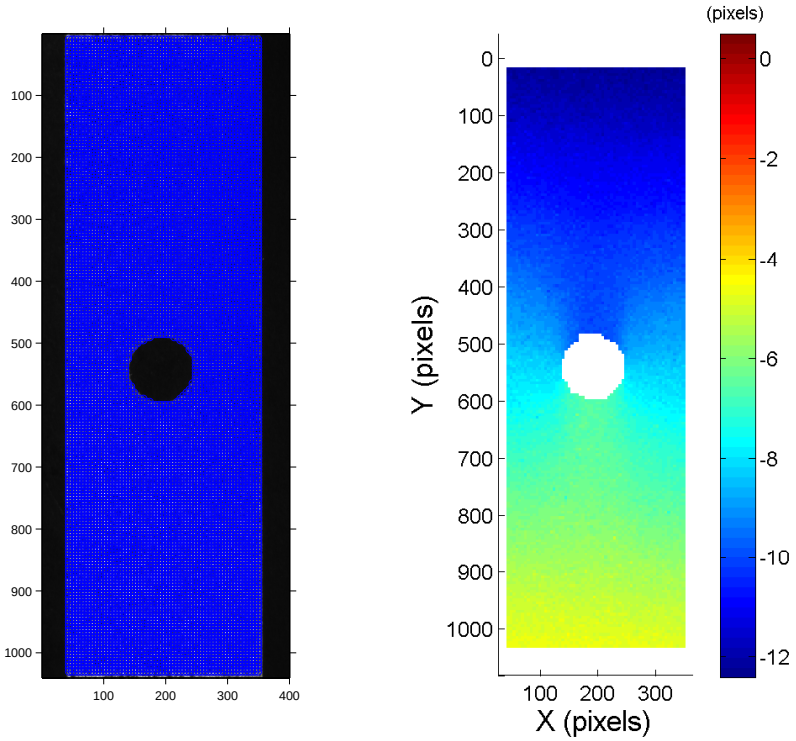


Figure 22: Options available to delete data. For this example, click on "Grid Points" and leave the image preview number set to 1.



(a) New grid

(b) V, with new grid

Figure 23: (a) New grid, after all the grid points that were not over the specimen (i.e. left and right sides of the image and the hole in the center of the specimen) have been deleted. (b) Contour plot of the vertical displacements for the last image (image 12) after extraneous grid points have been deleted.

7.7 Smooth Displacements and Calculate Strains

1. Smooth displacements to reduce noise and calculate strains.

   a) Run *compute_data_GUI*.

   b) (Fig. 24) For this example, leave the scale at 1 μm/pixel.

   c) Use the default smoothing kernel of 15 and smooth the displacements 3 times.

   d) Calculate strains using the 16-node, bi-cubic finite element interpolation scheme.

   e) Compute the deformed grid.

2. Visualize smoothed data and strains.

   a) Run *visualize_data_GUI*.

   b) Look at both the contour plots and line scans of the smoothed displacements and strains to determine if the smoothing parameters produced satisfactory results.

   c) If necessary, re-run *compute_data_GUI* and adjust smoothing parameters[2].

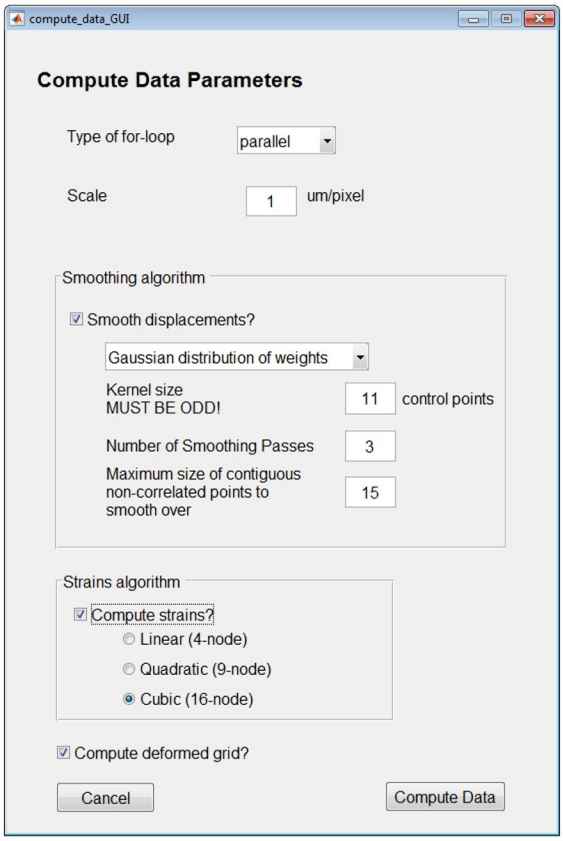
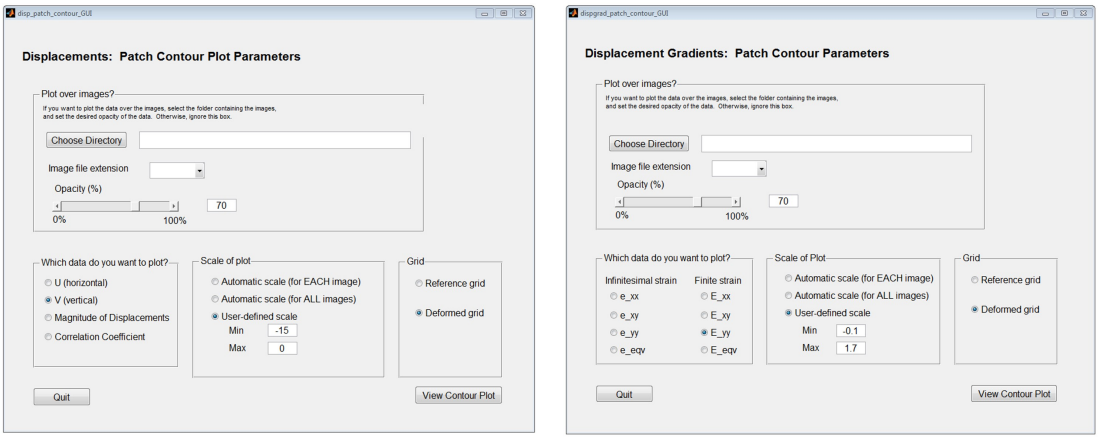


Figure 24: Options available for smoothing displacements and calculating strains.

[2]In order to determine the best smoothing parameters to use, run *compute_data_GUI* several times, using different kernel sizes and different numbers of smoothing passes. The correct amount of smoothing is subjective, and is a compromise between low noise in the strains and high spatial resolution of the strains.



(a) Contour Parameters for V

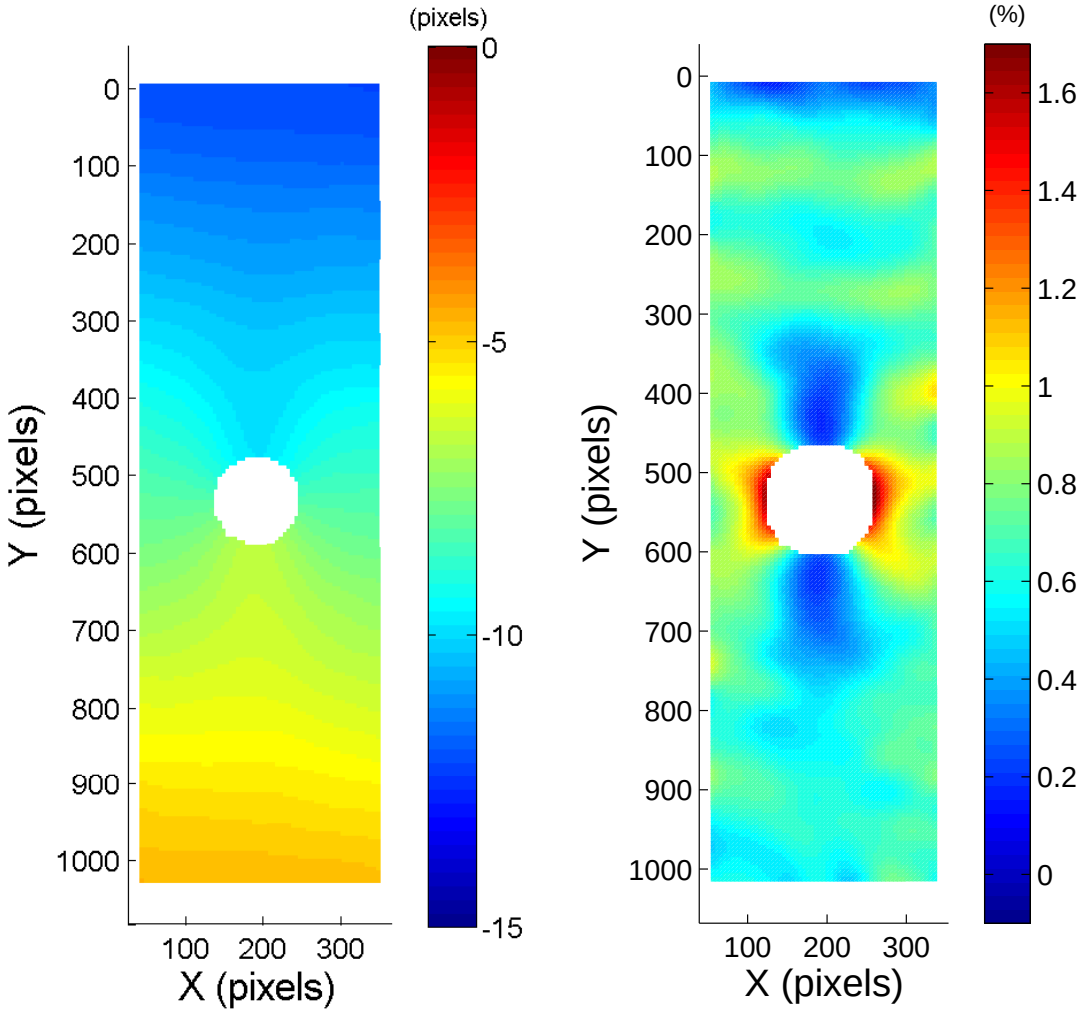
(b) Contour Parameters for E_{yy}

Figure 25: Settings used to generate contour plots of the vertical displacements, V (a) and vertical normal strains, E_{yy} (b) shown in Fig. 26.

7.8 Visualize Data

Visualize the data and save any desired plots.

1. Run *visualize_data_GUI*.
2. (Fig. 25) View the vertical displacements (V) and the vertical normal strains (E_{yy}) as contour plots and compare to Fig. 26.
3. The analytical solution for a hole in an infinite plate undergoing uniaxial tension shows that there should be a stress and strain concentration of three times the nominal stress and strain at the edge of the two edges of the hole perpendicular to the loading. In this example, the nominal strain, away from the hole, is approximately 0.6% strain, while the strain at the horizontal edges of the hole is approximately 1.6% strain. This is close to the expected strain concentration of a hole in an infinite plate.



(a) Vertical displacement, V

(b) Vertical normal strain, E_{yy}

Figure 26: Contour plots of vertical displacements, V (a), and vertical normal strians, E_{yy} (b) for image 12. These are the final results.

8 Detailed Explanation of Main GUIs

8.1 Image Setup

Table 1: Image Setup parameters

GUI name:	*image_setup_GUI*
Input arguments:	Images to be analyzed
Output arguments:	*filenamelist.mat*
	image_setup.mat
Parameters:	– File extension of images (i.e. .tiff, .tif, .jpg, etc)
	– Image skip

The purpose of this GUI is to prepare a list of the images the user wishes to correlate at once, saved as a Matlab variable *filenamelist.mat* in the current working directory. All images contained in this list will have as a reference image either a) the alphabetically first image in the folder or b) the preceeding image in the folder. (See Sec. 8.2.1 for more information concerning reference images.) If one wishes to correlate some images against one reference image, and other images against a different reference image, one must separate these two groups of images into two different folders, with the corresponding reference image as the first image in the folder, and run this code two times, once for each set of images.

The current working directory must contain all the images one wishes to correlate at once. The images must all have the same file extension, and they can be either black and white or color. They should be named such that their alphabetic order follows a logical sequence. The image names need not be the same length. Examples of acceptable naming conventions are found in Table 2.

Table 2: Examples of acceptable image name conventions

1.tif	Pic1.jpg	347900.734375.tiff
2.tif	Pic5.jpg	348802.156250.tiff
3.tif	Pic10.jpg	349703.171875.tiff

Though the image names need not be the same number of characters, the variable *filenamelist.mat* is a matrix, with each row corresponding to an image, and each column corresponding to a character in the image name. Therefore, to keep the matrix dimensions consistent, the *image_setup_GUI* finds the image with the longest name, and adds the appropriate number of zeros to the beginning of all image names that are shorter than the longest image name. This GUI will rename all the images in the folder. If you do not wish to have the images renamed, keep a copy of the original images in a different folder.

Though the original images can be in color, the correlation code requires black and white images. Therefore, the *image_setup_GUI* automatically converts all images in the folder to black and white if they are not already in this format. If you have color images and do not want to have the images converted to black and white, keep a copy of the original images in a different folder.

The parameter "Image Skip" allows the user to correlate only some of the images in the folder. If "Image Skip" is set to 3, for example, then images 1, 4, 7, etc... will be correlated. This is useful paricularly for the first couple of correlations, when the user wants to see the effect of various correlation parameters on the results, and does not need to see (or want to wait on) the results of every image.

The output of this GUI is the Matlab variable *filenamelist.mat*. Any method can be used to generate this variable. This GUI simply provides an automatized method that should work for most image naming conventions. The output *image_setup.mat* saves the parameters chosen when the *image_setup_GUI* was run.

8.2 Correlate Images

The purpose of this GUI is to define all the parameters for the correlation process. One must first have run *image_setup_GUI*, or have generated the variables *filenamelist.mat* and *image_setup.mat* in the appropriate format through some other means.

Table 3: Correlate Images parameters

GUI name:	*correlate_images_GUI*
Input arguments:	Images to be analyzed *image_setup.mat* *filenamelist.mat*
Output arguments:	*corr_setup.mat* *corr_setup_reduced.mat* *disp_initial_data.mat* *disp_raw_data.mat* *disp_reduced_data.mat* *grid_data.mat* *grid_reduced_data.mat* *grid_setup.mat* *grid_setup_reduced.mat* *valid_data.mat*
Parameters:	– Parallel or serial computation – Reference image (for serial computation only) – Correlation of reduced images (for large displacements) • Image Reduction Factor • Grid of control points • Subset Size • Threshold • Search Zone – Correlation of full-sized images • Use of data from reduced images • Grid of control points • Subset Size • Threshold • Search Zone

8.2.1 Serial/Parallel Computing and Reference Image

This code has the option of running in serial (each image is correlated sequentially, one at a time) or in parallel (multiple images are correlated independently at the same time on separate processor cores). Running the code in parallel is much faster than running it in serial for a large number of images. For only a few images, serial computing may be faster due to the overhead computation time associated with parallel computing. Serial computing is recommended if the user is editing the code as the code is easier to debug in serial mode.

The default reference image is the first image in the current working directory, which should also be the first image in the variable *filenamelist.mat*. However, this code allows for the option of using the preceding image as a reference image, which could be useful when dealing with large deformations and/or speckle patterns that become distorted during the specimen deformation process. If one uses this option, the cumulative displacements are calculated and reported with respect to the first image. Note that there is cumulative error associated with using the preceding image as a reference image. Therefore, the author recommends using the first image as a reference image when possible. See Sec. 8.2.3 for the recommended technique for correlating images with large displacements.

Because the par-for loop requires each loop iteration to be independent, when using parallel computation, the reference image MUST be the same for all the images to be correlated. To use the preceding image as the reference image, one must choose the serial computing option.

8.2.2 Grid Generator

A grid of control points at which one wishes to compute displacements must be defined. Though this grid can be generated by the user via any method he wishes, the function *grid_generator_GUI_compatible_3.m*, included within the *correlate_images_GUI*, provides a semi-automated method for generating the grid.

If a grid has not already been defined (either by the user, or via previous iteration of *correlate_images_GUI*), then *correlate_images_GUI* will prompt the user to define a grid using the default *grid_generator_GUI_compatible_3.m*. If a grid does already exist, then the user has the option of using the existing grid or defining a new grid. If the user defines a new grid, then the new grid will overwrite the old grid.

When the grid generator function *grid_generator_GUI_compatible_3.m* runs, the user is first prompted to open the reference / first image. Then the user must choose the region of interest (ROI) by selecting two points that form the diagonal corners of a rectangle. The user is then prompted to select a grid spacing. This value will define the number of pixels in between adjacent control points, both in the vertical and horizontal direction. The grid is then plotted over the image, and the user is given the option of accepting the grid as is, removing points, or starting over. If the user wishes to remove some grid points, he must select the diagonal corners of a rectangle surrounding the points he wishes to remove. He may choose to do this as many times as desired to remove points from different regions. If the user chooses to start over, the original grid is erased and the user must select a new ROI and a new grid spacing. When the user is satisfied with the grid, he must choose "Accept the grid." The code then proceeds directly to correlating the images.

Note that the original grid generator function, written by C. Eberl, contains many options for generating grids, for example circular grids with radial or circumferential points. The current author eliminated all options except for the rectangular grid option because the other original options are not compatible with the strain measurement code in *compute_data_GUI* (see Sec. 8.3.3).

8.2.3 Large Displacements and Reduced Images

The code has a maximum allowable vertical and horizontal displacement that it can find for any given control point, determined by the subset size and the search zone through Eqn. 1:

$$u_{max} = \frac{SS}{2}(SZ - 1) - 1 \tag{1}$$

where u_{max} is the maximum allowable vertical or horizontal displacement, SS is the subset size, and SZ is the search zone. The search zone is the area about the subset in which the code searches during the correlation. By default, $SZ = 2$. See Section 8.2.4 for more information on how the search zone is used during the correlation.

If the actual displacement at a control point is larger than the maximum allowed displacement, then the code will not correlate at that control point, and no displacement data will be returned. In order to obtain displacement data when the test specimen underwent large displacements, the user has two options: increase the search zone size or provide initial guesses for the displacements. Increasing the search zone size has two significant drawbacks. First, it drastically increases the computation time required for a correlation. Second, it requires a larger border between the outermost control points and the image edges. The recommended method for capturing large displacements is

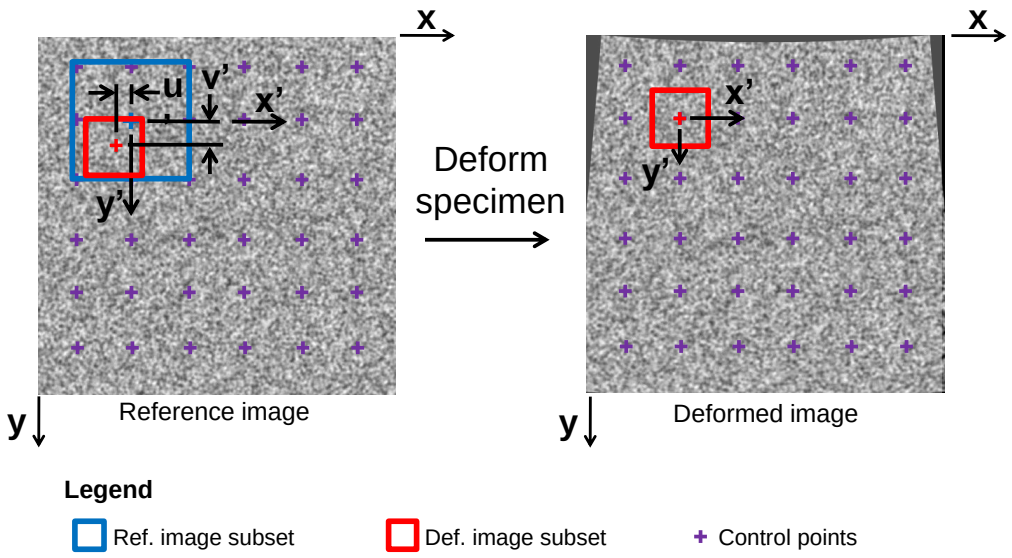


Figure 27: Schematic of digital image correlation methodology. A grid of control points (purple crosses) is defined over the region of interest. The normalized cross-correlation coefficient (Eqn. 2) is computed by convolving a subset in the deformed image (red box) with the corresponding larger subset in the reference image (blue box). The actual displacement (u,v) is the displacement that maximized the correlation coefficient.

to provide initial guesses for the displacements, which is described in the remainder of this section. In some extreme cases, however, the user will need to both provide initial guesses and increase the search zone. For this reason, Version 2 of the code allows the user to control the search zone value.

In order to provide initial guesses of the displacements in an automatic fashion, the user can correlate the reduced-size images. The code for correlating reduced images essentially shrinks the images by the amount chosen in "Image Reduction Factor," which serves to effectively reduce the size of displacements (in terms of pixels). The code then correlates the reduced images, and uses the results of the reduced correlation as initial guesses for the correlation of the full-size images.

The "Image Reduction Factor" should be set between 2 and 4. Reducing the images to less than 1/4 of their original size leads to poor correlation results. If a reduction of 4 is not sufficient to capture the large displacements, one can run the reduced image correlation iteratively. By selecting "Yes - use previous iteration" from the "Correlate reduced images?" drop down box, the code will use the results from the previous reduced correlation as initial guesses in the current reduced correlation.

It is recommended that the user obtain satisfactory reduced correlation results before running the full correlation, otherwise the adage "garbage in, garbage out" applies to the full correlation. To use the data from the reduced correlation as initial guesses in the full correlation, the user must select "Yes" under the "Use reduced data?" drop down box in the *correlate_images_GUI*.

8.2.4 Correlation of the images

The heart of the correlation process is in the *cpcorr_mod.m* function, a slightly modified version of the standard function in the Image Processing Toolbox, *cpcorr.m*, written by Matlab. For each control point, the code takes a subset of pixels around the control point in the deformed image, and a larger subset (generally twice as big) around the corresponding control point in the reference image, as shown in Fig. 27. The size of the reference subset with respect to the deformed subset is controlled by the search zone. Increasing the search zone will increase the maximum allowable displacement, but at the cost of a larger border around the region of interest in which control points cannot be correlated and increased computation time.

The code computes the normalized cross-correlation coefficient, C, [2] for a range of theoretical displacements, (u', v'), in 1 pixel increments by convolving the subset from the deformed image with the larger subset from the reference image according to:

$$C(u',v') = \frac{\sum_{x',y'} \left[\left(r(x',y') - \bar{r}_{u',v'} \right) \left(d(x'-u',y'-v') - \bar{d} \right) \right]}{\left\{ \sum_{x',y'} \left[\left(r(x',y') - \bar{r}_{u',v'} \right)^2 \right] \sum_{x',y'} \left[\left(d(x'-u',y'-v') - \bar{d} \right)^2 \right] \right\}^{1/2}} \tag{2}$$

where r is the intensity of the pixels in the reference subset, d is the intensity of the pixels in the deformed subset, and (x', y') are local subset coordinate axes whose origin is at the control point at the subset center. If the value of the computed correlation coefficient at a given control point is less than the value set for the threshold (typically 0.5), the correlation is determined to be poor and no data is returned for that control point. By decreasing the threshold value, the user can allow more grid points to be correlated, but at the expense of having less certainty in the validity of the correlations.

In order to calculate displacements to within 1/100 of a pixel, the nine discrete correlation coefficients surrounding the absolute maximum coefficient are interpolated using a second order polynomial in u' and v'. The actual displacement (u, v) for a control point is the theoretical displacement (u', v') corresponding to the maximum interpolated correlation coefficient. The output of the code is discrete displacement values for each control point for each image.

8.3 Compute Data

Table 4: Compute Data parameters

GUI name:	*compute_data_GUI*
Input arguments:	All output from previous GUIs
Output arguments:	*disp_smooth_data.mat* *DU_data.mat* *FEM_setup.mat* *disp_raw_data.mat* *grid_scale_data.mat* *smooth_setup.mat* *grid_deformed_data.mat*
Parameters:	– Parallel or serial computation – Scale (um/pixel) – Smoothing parameters • Even or Gaussian distribution of weights • Kernel size • Number of smoothing passes • Maximum size of uncorrelated points to smooth over – Strain computation parameters • Type of finite element used for interpolation of displacements – Computation of deformed grids

This GUI performs four main functions: it scales the displacement data from pixels to microns, smooths the displacement data, computes strains from the displacement data, and computes the deformed grids. These four functions are independent, but due to noise inherent in DIC displacements, it is strongly recommended that the user smooths the displacements before calculating strains and the deformed grid. The user can run this GUI multiple times in order to fine-tune the smoothing parameters before calculating strains. (The code uses the raw displacement data each time it runs.)

8.3.1 Scaling the displacements

The displacements from the *correlate_images_GUI* have pixels for units as default. In order to scale the data appropriately, the user must enter the pre-determined scale. The GUI, as set up, requires the physical units to be microns. However it is straight forward to edit the GUI and associated code in order to use a different unit that is appropriate to the user's length scale. If the user leaves the scale at 1 um/pixel, then all the displacement data is reported in pixels in the *visualize_data_GUI*.

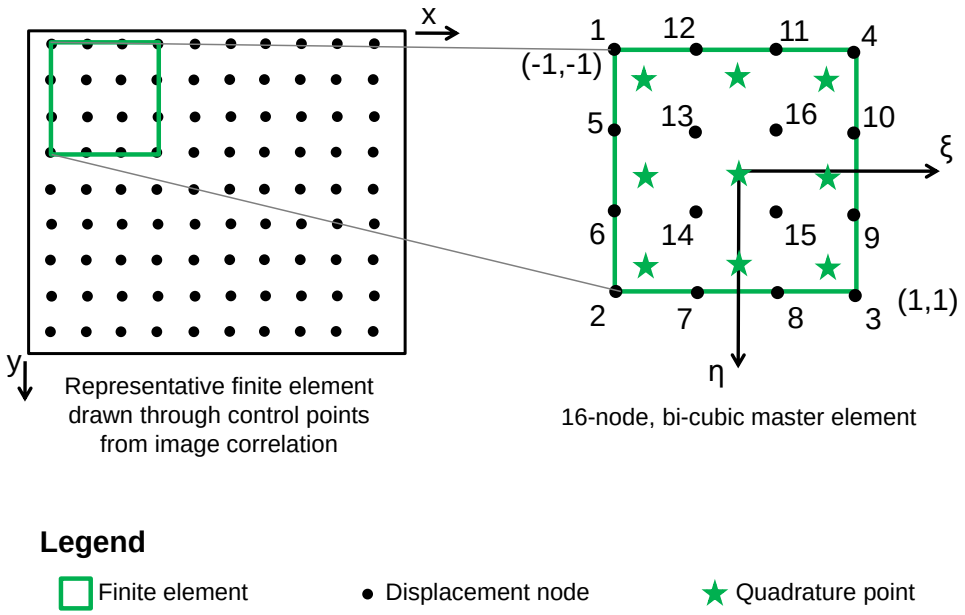


Figure 28: Schematic of finite element methodology used in strain calculations. Given displacements at a grid of control points (black circles), a 16-node finite element (green box) is drawn through the control points. This element is mapped to a master element, with local coordinates ξ and η, and the displacements are interpolated over the master element using bi-cubic finite element shape functions. The derivatives of the interpolated displacements are calculated at the nine Legendre-Gauss points of the element (green stars), and then mapped back to the original element.

8.3.2 Smoothing of displacements

Displacements are smoothed prior to calculating strains to provide robustness against the noise inherent to DIC. In order to reduce edge effects, the borders of the region of interest are first padded by reflecting displacement values at the edge of the region of interest across the border such that the first derivative is continuous. Version 2 of the code does this by using the built-in function *scatteredInterpolant*.

The user can select the kernel size of control points used in the smoothing process, as well as the weighting function. If "Even distribution of weights" is selected, then all the control points in the smoothing kernel are given equal weighting during the averaging. If "Gaussian distribution of weights" is selected, the displacements at control points within the kernel are weighted with a Gaussian distribution centered at the central control point and then averaged. The displacement at the central control point is replaced by the weighted average. To obtain smoother results, this process can repeated, by increasing the "Number of Smoothing Passes," whereby the averaged displacements from the previous smoothing iteration are used as inputs in the next smoothing iteration.

When an uncorrelated point is encountered, the code will extrapolate the displacements from neighboring control points to provide a displacement value for the uncorrelated point. For small numbers of contiguous uncorrelated points, this extrapolation process is reasonable; however, for large regions of uncorrelated points, the user may wish to not have the code extrapolate data over the region. The user can choose the maximum size of contiguous uncorrelated points for which the code will extrapolate the displacement data.

8.3.3 Strain Calculations

Strains are calculated by interpolating displacements using finite element shape functions.

A 16-node finite element is drawn such that the nodes of the finite element correspond to the control points from the image correlation. This element is then mapped to a master element, as shown in Fig. 28, through:

$$(x, y) = f^i(\xi, \eta) = \underline{\hat{N}}(\xi, \eta)\underline{X}^i \tag{3}$$

where (x, y) are the image coordinates, (ξ, η) are the local coordinates of the master element, and f^i is the mapping function, defined by the matrix multiplication of the finite element shape functions defined on the master element,

$\underline{\hat{N}}$, and the image coordinates of the nodes for element i, $\underline{X}^i$. The sixteen bi-cubic finite element shape functions, $\hat{N}_j$, are defined according to:

$$\hat{N}_j(\xi,\eta) = \left(a\xi^3 + b\xi^2 + c\xi + d\right)\left(e\eta^3 + f\eta^2 + g\eta + h\right) \tag{4a}$$

$$\hat{N}_j(\xi,\eta) = \begin{cases} 1 & \text{at node } j \\ 0 & \text{at all other nodes} \end{cases} \tag{4b}$$

where $a, b, ..., h$ are constants. Displacements are interpolated over the master element and mapped back to the original element through:

$$\underline{U}^I(x,y)|_{(x,y)=f^i(\xi,\eta)} = \underline{\hat{N}}(\xi,\eta)\underline{U}^i_{EL} \tag{5}$$

where $\underline{U}^I(x,y)$ are the interpolated displacements as a function of image coordinates (x,y) and $\underline{U}^i_{EL}$ is a matrix containing displacements in x and y directions of the element nodes for element i. The first partial derivatives, $\underline{\nabla U}(x,y)$, of the interpolated displacements are calculated through:

$$\underline{\nabla U}(x,y)|_{(x,y)=f^i(\xi,\eta)} = \underline{\nabla \hat{N}}(\xi,\eta)\left[\underline{J}^i(\xi,\eta)\right]^{-1}\underline{U}^i_{EL} \tag{6}$$

where $\underline{J}^i$ is the Jacobian of the mapping function for element i, defined according to:

$$\underline{J}^i(\xi,\eta) = \left[f^i(\xi,\eta)\right]' = \underline{\nabla \hat{N}}(\xi,\eta)\underline{X}^i \tag{7}$$

The interpolation scheme is most accurate at the nine Legendre-Gauss points of the master element [1, 3], defined by combinations of $(\xi,\eta) = \{-\sqrt{3/5}, 0, \sqrt{3/5}\}$. The displacement derivatives are therefore calculated at these discrete points for all possible elements that can be drawn through the control points. Overlapping Gauss points from neighboring elements are then averaged, giving discrete displacement gradient values on a grid slightly smaller than the grid of control points. Components of the small-strain tensor, ϵ_{ij}, and the Green-Lagrangian finite strain tensor, E_{ij}, are then calculated:

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial X_j} + \frac{\partial U_j}{\partial X_i}\right) \tag{8a}$$

$$E_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial X_j} + \frac{\partial U_j}{\partial X_i} + \frac{\partial U_k}{\partial X_i}\frac{\partial U_k}{\partial X_j}\right) \tag{8b}$$

A rotationally-invariant equivalent strain is included in the code, in order define a state of strain that is independent of the orientation of the test specimen with respect to the image. This definition is based on the Von Mises equivalent stress, and is defined as:

$$\begin{aligned} E_{eqv} &= \left[\frac{3}{2}\left(E_{ij}E_{ij} - \frac{1}{3}E_{mm}E_{mm}\right)\right]^{1/2} \\ &= \left[\frac{3}{2}\left(E_{11}^2 + E_{12}^2 + E_{21}^2 + E_{22}^2 - \frac{1}{3}\left(E_{11}+E_{22}\right)\left(E_{11}+E_{22}\right)\right)\right]^{1/2} \\ &= \left[E_{11}^2 + E_{22}^2 - E_{11}E_{22} + \frac{3}{2}\left(E_{12}^2 + E_{21}^2\right)\right]^{1/2} \end{aligned} \tag{9}$$

Note that this definition of equivalent strain is invariant to rotations, shown in Eqns. 10 & 11, where Q_{ri} is a rotation matrix, and prime notation denotes tensor components in the rotated coordinate system. Note that $E'_{kk} = E_{kk}$ because the trace of tensors is rotation invariant.

$$E'_{rs} = Q_{ri}Q_{sj}E_{ij} \tag{10}$$

$$
\begin{aligned}
E'_{eqv} &= \left[\frac{3}{2}\left(E'_{rs}E'_{rs} - \frac{1}{3}E'_{mm}E'_{kk}\right)\right]^{1/2} \\
&= \left[\frac{3}{2}\left(Q_{ri}Q_{sj}E_{ij}Q_{ro}Q_{sp}E_{op} - \frac{1}{3}E_{mm}E_{kk}\right)\right]^{1/2} \\
&= \left[\frac{3}{2}\left(\delta_{io}\delta_{jp}E_{ij}E_{op} - \frac{1}{3}E_{mm}E_{kk}\right)\right]^{1/2} \\
&= \left[\frac{3}{2}\left(E_{ij}E_{ij} - \frac{1}{3}E_{mm}E_{kk}\right)\right]^{1/2} \\
&= E_{eqv}
\end{aligned} \tag{11}
$$

8.3.4 Computation of the Deformed Grid

The user can choose to compute the deformed grids for both displacements and strains. For displacements, this function simply adjusts the reference grid control points based on the smoothed displacements (or raw displacements, if displacements are not smoothed) for each image correlated. For the strains, the function first interpolates the displacement field to obtain displacement values for at the control points in the strain reference grid. (Recall that the strain results are presented on a slightly different grid than the displacements.) Then, the strain reference grid control points are shifted by the interpolated displacement values.

If the user computes the deformed grids, then the user has the option of plotting the data in *visualize_data_GUI* over either the reference grid or the deformed grid. Note that the deformed grid is only calculated for correlation of the full images; it is not calculated for correlation results of reduced images.

8.4 Visualize Data

Table 5: Visualize Data parameters

GUI name:	*compute_data_GUI*
Input arguments:	Images to be analyzed All output from previous GUIs
Output arguments:	Graphs and Plots of data
Parameters:	– Reduced vs Full Correlation Results – Raw or Smoothed Displacements – Type of plot: • Vector field (displacements only) • Contour plot • Line scan • Spatial average

There are four default methods of visualizing the displacement and strain data: as a vector field (displacements only), a filled contour plot, a line scan, or a spatial average. The first three options plot the data for one image at a time. To advance through the data from all of the correlated images, the user clicks on the figure. At any point, the user can save a Matlab figure and then continue to advance. If the user wishes to generate other types of plots, he could easily incorporate them into the *visualize_data_GUI*. The author requests that if a user expands the visualization options, he notifies the author so that the additional capabilities can be incorporated into a future version of the code.

8.4.1 Vector Field

This function uses Matlab's *quiver.m* to generate a vector field of displacements. The user can adjust the scale of the arrows used in the plot via the "Quiver Scale" parameter. Matlab scales the arrows by first scaling them such that they fit within the grid, and then stretching them by a factor of "Quiver Scale". See `http://www.mathworks.com/help/matlab/ref/quiver.html` for more information.

As DIC data is often very dense (i.e. control points every 5-10 pixels), the user may choose to reduce the number of arrows plotted through the "Quiver Skip" parameter. If the "Quiver Skip" is set to 10, then every 10th quiver will be plotted.

8.4.2 Filled Contour Plot

This function uses Matlab's *patch.m* function to generate a filled contour plot of displacements and strains. The function generates a square patch around each control point and adjusts the color of the patch based on the value of the data at that control point. By plotting all of the patches together, a filled contour plot is generated. When plotting a contour plot over the deformed grid, which is no longer regularly spaced, the patches are no longer contiguous and may either over lap and/or have a gap between them. This behavior is a known deficiency of the code, and the author welcomes any suggestions for improvement.

Of note is the absence of a line contour plot using Matlab's *contour.m* function. The *contour.m* requires an evenly-spaced, rectangular grid over which to plot the data. The grid from the image correlation, however, is often not strictly rectangular because e.g. the region of interest may not be rectangular, there may be a hole in the specimen, regions of data that did not correlate well may be deleted, etc. Additionally, the *contour.m* function would not allow the user to plot a contour plot over the deformed grid. Using the *contour.m* function with a non-rectangular or unevenly-spaced grid would require interpolating data over an evenly-spaced, rectangular grid, which could lead to errors from the interpolation. For these reasons, the author decided to have contour plots generated through the *patch.m* function.

The user can choose to plot the contour by itself, or can plot the contour semi-transparently over the corresponding image in the background. To plot a simple contour plot without the images, simply leave the "Choose Directory" box blank or enter a zero in the box. To plot over a background image, select the folder containing the images, and select how opaque the data should be. The folder containing the images does not necessarily need to be the same folder that contains the correlated data, but it should contain the exact same images that were used in the correlation. Note, in order to see x- and y-coordinate tick marks when plotting a contour plot over a background image, you must set the tick mark preference by using the command *iptsetpref('ImshowAxesVisible','on')*. You only need to type this command in the command window one time, and the preference will be set permanently.

The user can also choose the scale desired for the contour plot color bar. The option "Automatic scale (for EACH image)" automatically calculates the minimum and maximum of the appropriate data for each image. This option will allow the color bar to be scaled for EACH image. The second option, "Automatic scale (for ALL images)"calculates the minimum and maximum of the appropriate data for all images. This option will keep the color bar constant for all images. As the third option, the user can specify the scale to uses; this option keeps the color bar constant for all images.

8.4.3 Line Scan

This function plots data along a line scan in the region of interest. The user can select either a vertical or a horizontal line at any location within the region of interest (ROI). For example, if the "Fraction of ROI" is set to 0.1, and the "Direction of line scan" is vertical, the line scan will be along a vertical line close to the left side of the ROI. If "Fraction of ROI" is 0.9 and "Direction of line scan" is horizontal, the line scan will be along a horizontal line near the bottom of the ROI.

8.4.4 Spatial Average

This function plots an average of the data over either the entire region of interest (ROI), over a certain line, or over a rectangular region that is a fraction of the ROI and is centered with the image center. The user can choose to plot the averaged data against either image number or time on the x-axis.

8.5 Delete Poorly Correlated Data

It is not uncommon in DIC to have portions of the region of interest that do not correlate well because, e.g. the test specimen rotated and the image went out of focus or the speckle pattern was poor. In these instances, it can be advantageous to delete the poorly correlated data and have a smaller region of interest with good data.

In principle, the user could either (a) delete the raw displacement data first, and then smooth the data and compute strains, or (b) smooth all of the data and compute strains and then delete the poorly correlated regions. Both methods have their advantages and their drawbacks. A disadvantage of (b) is that the strains near the poorly-correlated region are affected by the poorly correlated displacements. Also, the smoothing function interpolates data

at control points that did not correlate at all, making it more difficult to determine the poorly-correlated region accurately. In contrast, method (a) allows the user to delete poorly correlated data before strains are calculated, so that the poor displacement data does not affect the strains. However, a disadvantage of (a) is that edge effects are introduced during the smoothing process along edges of deleted data.

It is the author's opinion that method (a) is preferred, and therefore the *delete_data_GUI* must be used with only raw displacement data; smoothed data and strains must be computed after the desired raw data is deleted. The author is currently working on reducing edge effects during the smoothing process, and would gladly accept suggestions from users.

8.6 Create Movies

The *movie_GUI* creates time-lapsed movies of the contour data for either displacements or strains. The user can control the frame rate and whether or not to plot the plain contour plot or plot over a background image.

To create the movie, the user firsts previews a representative figure (determined by the *Preview Image Number*, and here, the user can adjust the figure parameters (e.g. title). When creating the movie, it is important that the user not cover the Matlab figure with the mouse or other windows. Otherwise, the windows covering the figure will be captured in the movie.

The function currently is set to make MPEG-4 movies. However, the user can modify these options in line 45 of *patch_contour_movie*. See `http:/?www.mathworks.com/help/matlab/ref/videowriterclass.html` for more information.

At this time, the author has only written code for making movies of contour plots, but the user may wish to make movies of the changing vector field, changing line scan, etc. The author requests that if a user expands the movie options, he notifies the author so that the additional capabilities can be included in future versions of this code.

9 Performance of Code

The accuracy and precision of this DIC code is of extreme importance to all users. The author is currently participating in the DIC Challenge, a round-robin style competition sponsored by the Society of Experimental Mechanics (SEM) with the purpose of providing a set of standard images and evaluation criteria for all university and professional DIC codes and software. See `http://www.sem.org/dic-challenge/` for more information. A complete evaluation of this DIC code using the DIC Challenge image sets is provided in the PDF document "DIC Challenge - Summary of Results."

9.1 Known Deficiencies

There are three main areas of this code that the author currently recognizes as needing further improvement. First, during the smoothing of displacements, edge effects can be significant. To reduce edge effects, the code first pads displacements around the border of the ROI before smoothing them, but this does not eliminate the edge effects. Additionally, when control points are deleted, either before correlation using the *grid_generator_GUI_compatible_3.m* function, or after correlation using the *delete_data.m* function, the displacements are not padded on edges where data was deleted. Therefore, edge effects are even more pronounced on these edges. The author welcomes any suggestions for improved smoothing algorithms that have smaller edge effects.

Second, based on results of the DIC Challenge evaluation, the author found that this code does not capture large rotations well. See the accompanying PDF document "DIC Challenge - Summary of Results" for more information. At this time, the author does not have plans to improve the ability of the code to capture rigid rotations. If any user wishes to work on improving this aspect of the code, the author would be happy to discuss possible solutions.

Third, the filled contour plot generated by the *patch.m* function has a couple of drawbacks. Namely, the use of individual colored patches to represent the displacement or strain value at each control point results in non-smooth contours when plotting over the deformed grid. The author welcomes any suggestions for improving the contour plot option that works with both the undeformed and deformed grids and with semi-transparent data.

10 Concluding Remarks

The author is pleased to present this free, open-source DIC software to the general Matlab community. The code is constantly evolving as improvements are made, but the set of Matlab files included in this version represent a fairly complete, working DIC code.

Part of the appeal of this code, compared to commercial "black box" codes, is the ease of modification. As you use this code, please notify the author of any modification you make - either improvements or adaptations for a specific situation - so that such modifications can be addressed in future versions.

Contact information:
Elizabeth Jones
NSF Graduate Research Fellow
Graduate Research Assistant
University of Illinois at Urbana-Champaign
ElizabethMCJones@gmail.com

References

[1] John Barlow. Optimal stress locations in finite element models. *Int. J. Num. Methods Eng.*, 10:243–251, 1976.

[2] Mathworks. Normalized 2-D cross-correlation (normxcorr2), 2012.

[3] D. A. Tortorelli. *Solid mechanics : analysis and design with the finite element method.* Electronic Publication, Urbana, IL USA, 2010.