MultiDIC

Instruction Manual

Version 1.1.0

December 20, 2021

Dana Solav (danas@technion.ac.il)

Technion - Israel Institute of Technology

Table of Contents

1. Overview

MultiDIC (Multi Digital Image Correlation) is an open-source MATLAB toolbox by Dana Solav. Three-dimensional (stereo) Digital Image Correlation (3D-DIC) is an important technique for measuring the mechanical behavior of materials. MultiDIC was developed to allow fast calibration even for a large number of cameras, and be easily adaptable to different experimental requirements. It integrates the 2D-DIC subset-based software Ncorr with several camera calibration algorithms to reconstruct 3D surfaces from multiple stereo image pairs. Moreover, it contains algorithms for merging multiple surfaces, and for computing and plotting displacement, deformation and strain measures. High-level scripts allow users to perform 3D-DIC analyses with minimal interaction with MATLAB syntax, while proficient MATLAB users can use stand-alone functions and data-structures to write custom scripts for specific experimental requirements. Comprehensive documentation, user guide, and sample data are included.

2. Installation

2.1 Operating system requirements

MultiDIC was developed on 64-bit Windows 10 and has not yet been tested on other operating systems.

2.2 MATLAB requirements

This version of MultiDIC was developed on MATLAB versions 2017a and 2017b and has been tested on R2021b.

Matlab toolbox requirements:

- Simulink
- Image Processing Toolbox
- Statistics and Machine Learning Toolbox
- Computer Vision System Toolbox

2.3 MEX Setup

MultiDIC includes the 2D-DIC software **Ncorr**, which requires a MEX (C++) compiler. More details can be found in the Ncorr instruction manual, which is included in DuoDIC\lib_ext\ncorr_2D_matlab-master, or in the following links: http://www.ncorr.com/ and https://github.com/justinblaber/ncorr_2D_matlab.

First, you need a supported C++ compiler is installed on your computer. Find a C++ compiler that is supported for your MATLAB version here: https://www.mathworks.com/support/requirements/supported-compilers.html. If you're not using the most current MATLAB version, click on 'View compiler support for previous releases' on the top right side of the page to access the correct version.

After downloading a supported compiler, make sure MEX has been set up properly in MATLAB. Type `mex -setup` in the MATLAB Command Window. The output should be something similar to:

```
MEX configured to use 'MinGW64 Compiler (C++)' for C++ language compilation.
```

(The compiler name 'MinGW64' may be replaced by the name of your installed compiler).

The free MinGW C++ compiler, which is available for download in the above link, works as long as you only use the single threaded version of Ncorr. If you wish to use Ncorr's multithreading option, you will require a compiler that supports OpenMP (see Ncorr manual for further details).

2.4 Additional included packages

DuoDIC uses functions from **GIBBON:** the Geometry and Image-Based Bioengineering add-On (https://www.gibboncode.org/). All the necessary functions from GibbonCode are already included in DuoDIC; however, you are encouraged to check out what other capabilities GIBBON has to offer (finite element analysis, meshing tools, image segmentation, and more). DuoDIC also uses the following packages and codes: export_fig, findjobj, inputsdlg, numsubplots, selectdata, subtightplot, uipickfiles.

2.5 Installation

After MEX is set up correctly, in Matlab, navigate to the directory where you saved MultiDIC, and type `installMultiDIC` in the MATLAB terminal. This will compile all the necessary files for Ncorr and will save the Matlab MEX files in the Ncorr folder. It will also add all the necessary files to MATLAB search path. It has to be done only the first time. If this is the first time you install, you will get a dialogue box warning you about missing compiled files. Click OK to continue. Then, another dialogue box will appear regarding OpenMP support, as shown in Figure 1:
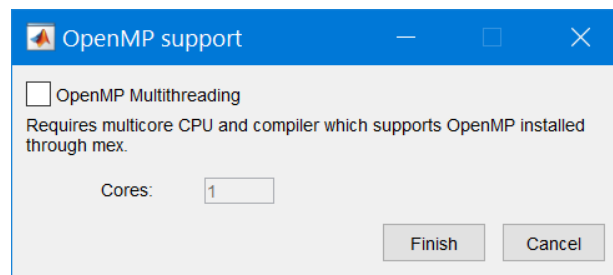

Figure 1. OpenMP support MATLAB dialogue box.

Click the checkbox for OpenMP if you have a multicore processor, want multicore support, and have installed a supported compiler with OpenMP support. Enter the number of cores on your system (which can be determined by checking the Performance tab in Windows' task manager. If you choose not to use multithreading, then just leave OpenMP support unchecked and click finish. The files should compile and display in the Command Window, similarly to what is shown below:

```
MEX completed successfully.
Installing ncorr_alg_dispgrad... Please wait
Building with 'MinGW64 Compiler (C++)'.
```

If all the files compiled properly, a Graphic User Interface (GUI) for Ncorr should appear, as shown in Figure 2. If this is the case, Ncorr has most likely installed correctly. The next time you open MATLAB, you will not have to repeat the installation process, as long as you do not delete the compiled MEX files that were saved in the Ncorr folder.
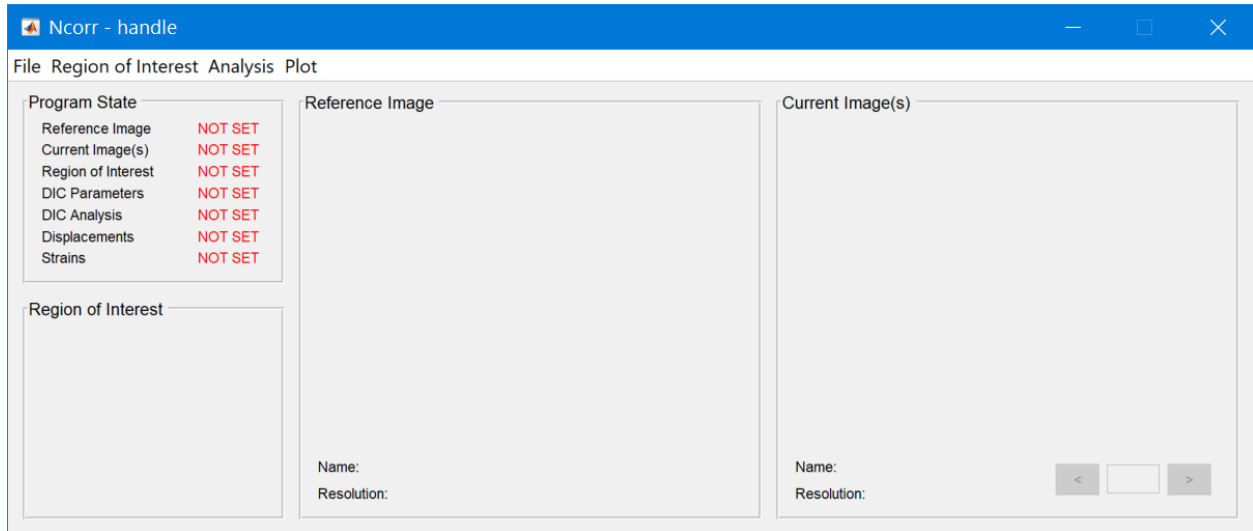
Figure 2. Ncorr MATLAB Graphic User Interface (GUI)

3. Preparation

The following items are required to complete a 3D-DIC analysis:

- A set of stereo calibration images, in which a 3D calibration object is imaged from all the cameras.
- A set of speckle images, in which the speckle test object is imaged by all cameras in a reference (e.g. undeformed) configuration and optionally also deformed configurations.
- A set of flat checkerboard images, necessary only in case distortion correction is required.

The preparation steps required for each of these steps are described in detailed in the following sections.

3.1 Calibration Objects

3.1.1 Stereo calibration object

Currently, the toolbox and the documentation are designed for a cylindrical or semi-cylindrical calibration object (see for example Figure 3) with black control points over white background. In order to calibrate a stereo pair of cameras, both cameras need to view an overlapping region of the calibration target, which contains at least 6 control points in their field of view. However, it is recommended that a much larger number of points is visible, in order to increase the calibration accuracy. For example, in Figure 3, camera 1 and camera 2, each have 200 points in their field of view, 120 of them are mutual for both cameras. Any number of stereo pairs can be positioned around the cylinder, as long as each pair has a valid field of view. It is not required to use the entire 360° around the cylinder; a cylindrical calibration object can be used also for reconstructing only certain portions of the space.

The points on the calibration object are arranged in N_r rows and N_c columns. It is recommended to include the column number above/below each column (See Figure 3). The dots can be square, circular, or any other shape, as long as their centroid coordinates are known with sufficient accuracy. A MAT file has to be created, containing a $N_r - by - N_c - by - 3$ array of the 3D world coordinates of the calibration target arranged by $[rows, columns, xyz]$, where z is the vertical coordinate, the rows are arranged from bottom to top (increasing z value) and the columns arranged counter-clockwise, such that the column number increases from left to right on the image. An example for such file can be found in the sample data. In addition, in case you are not sure how to create the 3D positions file for your cylindrical object, you can use the following script, which is included in the lib_MultiDIC folder: `generate_cylindrical_calibration_object_coordinate_file`.
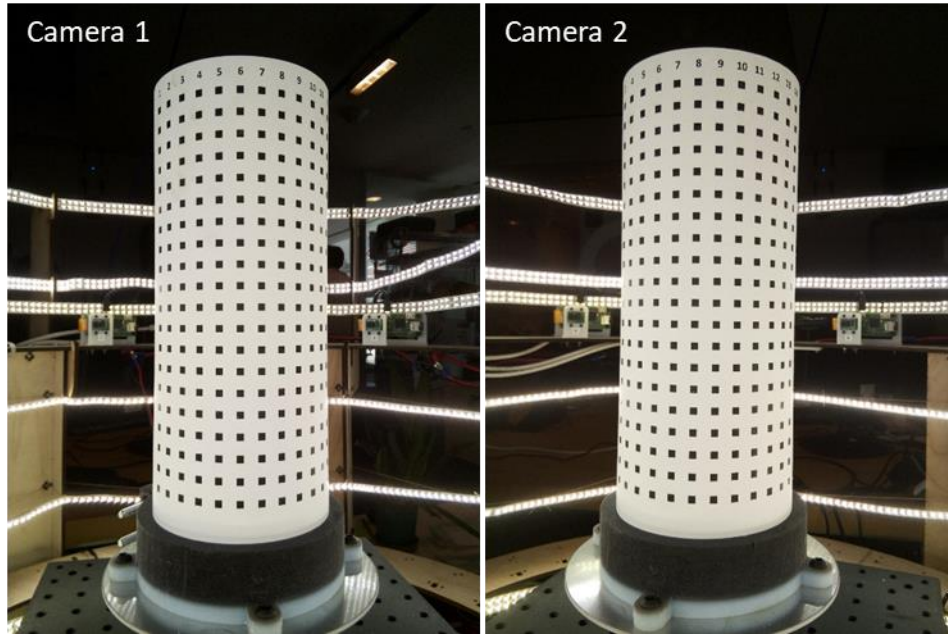
5

Figure 3. Calibration object prepared by applying a sticker paper with a printed dot pattern on an aluminum cylinder.

3.1.2 Flat checkerboard

- This is only necessary if you wish to include distortion correction.
- For most applications it is sufficient to use a flat calibration target made by attaching a pattern printed on adhesive paper onto a flat surface (such as a glass frame).
- Print an asymmetric checkerboard image (odd number of rows and even number of columns, for example) with (accurately) known square size.
- To print MATLAB's standard pattern, type open checkerboardPattern.pdf in the command window to print MATLAB's pattern.
- If you want to print a different size pattern (for example a different number of squares), you can use the MultiDIC function createCheckerBoardImage to create an image for printing. To do this, type
  createCheckerBoardImage(Nrows, Ncol, squareSize, resolution, fileName)
  in the command window, where Nrows and Ncol are your selected number of rows (odd number) and columns (even number), squareSize is the size of each square in meters, resolution is the required image resolution in pixels/meter (for example 600 dpi is equivalent to ~23622 pixels/meter), fileName is the name of the file for saving the checkerboard image (including the path directory unless you want it to be saved in the current directory).
- Attach the image on a flat surface (as perfectly flat as possible, a simple solution is printing on a sticker paper and sticking it on a glass picture frame). See for example Figure 4.
- Take a few dozen (at least 20, preferably closer to 50) images with the checkerboard positioned in different distances from the camera and tilted different angles, covering the entire field of view of the camera. Delete images where the board is cut (not entirely visible in the image), blurry/unfocused images, etc. It is not mandatory to do so, as the function calculateCBcalibrationParameters will automatically discard those images, but it will take more time to run.
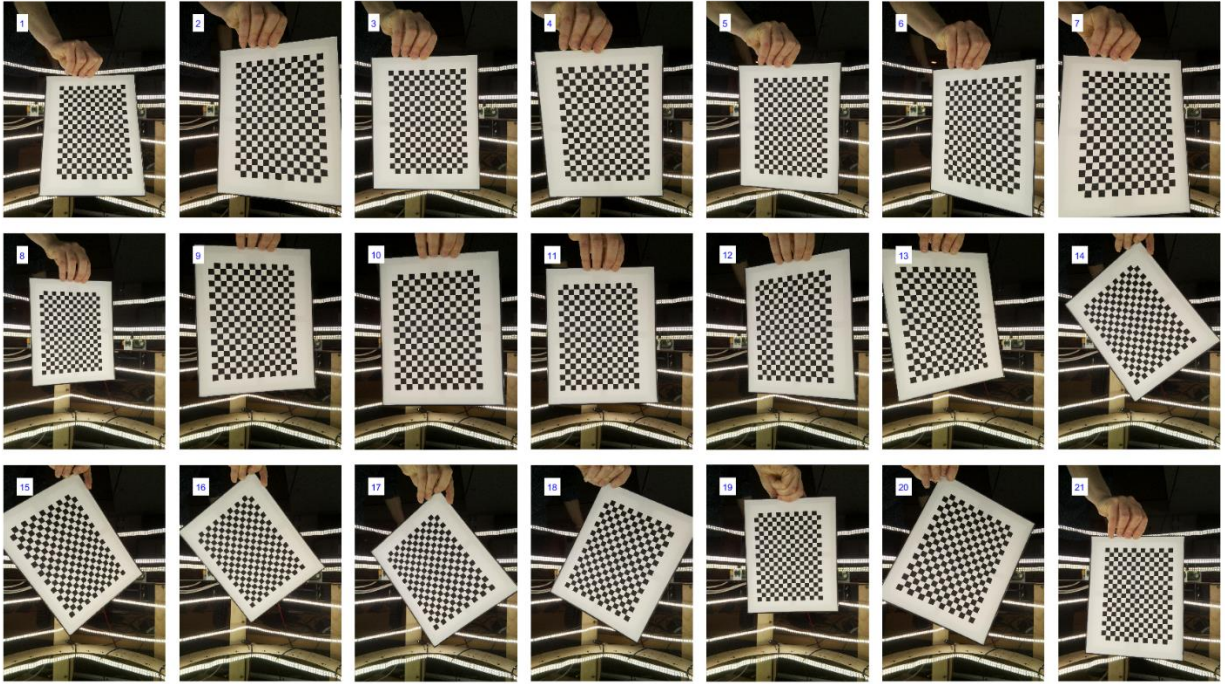
Figure 4. Example images of a checkerboard calibration target.

3.2 File Naming Convention

3.2.1 Stereo calibration images

Name each image with a name equal to the camera number or ending with the camera number after an underscore. For example, if the camera number is 101, the image file names can be: 101.jpg, CalibrationImage_101.jpg, IM_101.gif, Cam_101, etc.

3.2.2 Checkerboard images

Save the images taken by each camera in a separate folder. The folder name must contain the camera number after an underscore (for example "2" or "Cam_02" or "C_2" or "02", but not "cam2"). The image names do not matter.

3.2.3 Speckle images

Save the images from each camera in a separate folder. The folder name must be either the camera number or the camera number at the end after an underscore (for example "cam_02", "camera_7", "007" or "9", but not "cam05" or "cam_2_left"). The camera numbers must match those of the calibration steps. Inside each folder, the images should be named with the time frame after an underscore, in a way that will determine their order (For example cam01image_001, cam01image_002… or im_01, im_03…). Image number 1 is always considered as the reference image. The rest of the images will be ordered according to their numbers but they do not have to be consecutive. Note that MATLAB treats the numbers digit by digit, so you should pad them with zeros. For example: 1,2,3…10 will be ordered: 1,10,2,3,4… To prevent that from happening, name the images: im_01, im_02,… im_10, etc.

3.3 Speckling

As a rule of thumb, speckles should be at least 3-5 pixels in size, have good contrast, equal size black and white areas, and no directionality (random pattern) [1]. Various methods for applying the speckles onto the surface exist, such as spray painting, stamping, printing, airbrushing, etc. A recent review on speckle pattern fabrication can assist you to select a preferable method, according to your application [2].

3.4   Viewing and saving 3D figures

This toolbox uses GibbonCode's functions for plotting 3D points and meshes. It adds the vcw (View Control Widget) which allows users to better manipulate a view in 3D. Click the button or type v to activate it. If you re-open an existing figure, and the widget doesn't appear, type `vcw` in the command window to enable it. The widget allows the user to rotate, pan and zoom a figure using key presses and mouse gestures (right mouse button for zoom, left for panning, and middle/scroll for rotating). Press i to show help information:
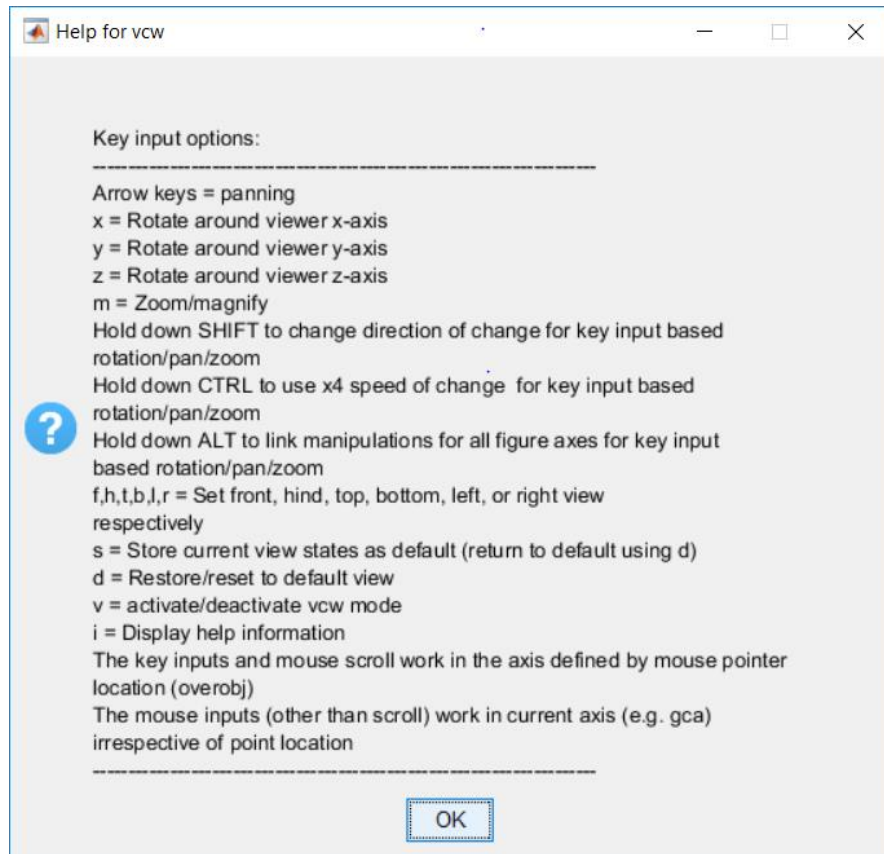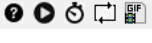

Figure 5. View Control Widget input options

The following are brief descriptions of a set of widgets that may also be useful in viewing and analyzing both 2D and 3D figures. When activated, each widget will be applied to the current figure by default, unless a different figure is otherwise specified in the function call.

- Colorbar
  - Click the button to activate the colorbar widget. This widget allows the user to view the current colorbar axis minimum and maximum and change these values to better reflect differences in plotting parameters across a figure.
- Colormap
  - Click the button to activate the colormap widget. This widget allows the user to select a new color scheme for the figure from a drop-down menu.
- Edge Color
  - Click the button to activate the edge color widget. This widget allows the user to remove or change the edge color of lines displayed on a figure.

- Face Alpha
    - Click the ⛫ button to activate the face alpha widget. This widget allows the user to view the current face alpha setting and change it to any value [0,1]. Greater values will make the figure opaque, while lesser values will make the figure more transparent.
- Light
    - Click the ☀ button to activate the light widget. This widget allows the user to project light onto a figure from the left, right, or center of the current display. The intensity of this light can be adjusted via the ambient strength widget.
- Ambient Strength
    - Click the ⬤ button to activate the ambient strength widget. This widget allows the user to view the current ambient strength setting and change it to any value [0,1]. Greater values will project more light all over the figure face, while lesser values will result in higher contrast and more concentrated light. The function of this widget must be coupled with the existence of light patches on a given axis (see Light).
- Diffuse Strength
    - Click the ☼ button to activate the diffuse strength widget. This widget allows the user to view the current setting and change it to any value [0,1].
- Specular Strength
    - Click the ✔ button to activate the specular strength widget. This widget allows the user to view the current setting and change it to any value [0,1].
- Face lighting algorithm
- Click the ⬢ button to activate the face lighting algorithm widget. This widget allows the user to view the current setting and change it to any of the following options: flat (uniform lighting across each face), gouraud (linear interpolation of the light across each face), and none (turn off lighting).
- Quiver
    - Click the ➚ button to activate the quiver factor widget. This widget allows the user to view the current quiver scale factor and adjust it as needed.

Moreover, clicking the 🖶 links to the `export_fig` function where users can specify file names, formats, and more. For more details: http://www.mathworks.com/matlabcentral/fileexchange/23629-export-fig.

Furthermore, in the animated figures, another set of widgets will appear on the window: ❓ ▶ ⏱ ↻ 🖼 . Use these buttons to play or scroll through the plots, change playing speed or cycle, and export animated .gif files.

4. User Guide

4.1 Program Flow

The workflow of MultiDIC is composed of 5 steps, each of them is executed using a main script. STEP0 is optional, and is only necessary in cases where the cameras' lens distortion is not negligible. STEP1 and STEP2 are both necessary, but the order in which they are performed is not important. STEP2 involves matching 2D points from stereo speckle image pairs while the imaged object is moving and/or deforming. STEP1 involves finding the calibration parameters for transforming these pairs of corresponding 2D image points into 3D world points. These parameters are calculated by analyzing stereo images of a 3D calibration object. STEP 1 can be done before or after the speckled object images are analyzed, and the calibration results from STEP1 can be used for more than one dynamic analysis, as long as all the optical settings are identical. Next, STEP3 uses the results from STEP1 (stereo calibration parameters for each camera) and of STEP2 (corresponding speckle image points), and optionally also STEP 0 (distortion parameters for each camera), and calculates the resultant 3D points for each pair of cameras. In

addition, in STEP3 a triangular mesh is calculated, as well as the displacements, deformations, strains, and rigid-body motion of the object. The workflow is illustrated in Figure 6. The user guide provides all the necessary instructions for running the scripts, without interacting with the codes themselves and with MATLAB syntax. For a deeper look into the codes, and ways to modify refer to Section 5.
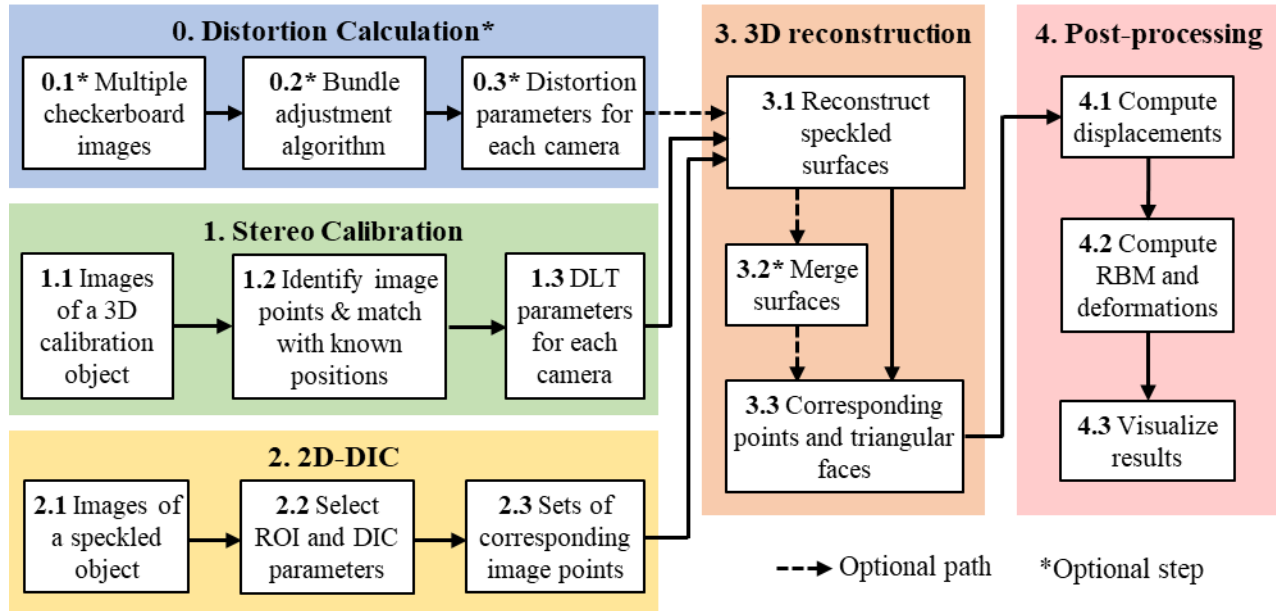
Figure 6. Software workflow

4.2   Projects in MultiDIC

When you perform a complete 3D-DIC analysis, the output files are stored and can be used multiple times. For example, once calibration parameters are calculated and saved, they can be used in multiple analyses of speckle images, as long as the cameras were not moved. Moreover, once distortion parameter calculation was performed, they can be used for multiple tests, as long as the cameras' intrinsics have not been changed.

4.3   Step 0: Calculate Distortion Parameters

STEP0_CalcDistortionParameters is the main script to calculate the distortion parameters of the cameras. The script uses camera calibration functions from MATLAB Computer Vision System Toolbox [3], which is based on the works of Zhang [4], Heikkila and Silven [5], Bouguet [6], and Bradski and Kaehler [7].

The script perform the following main steps:

1. Estimate camera parameters from multiple checkerboard images.
2. Use these parameters to correct for image distortion.
3. Plot the camera parameters and reprojection errors, before and after the correction.

The distortion parameters found in this step can be used to correct the distortion on all the images taken by the same camera, as long as the intrinsic parameters are left unchanged (the camera can be moved but the focus should not be changed). Specifically, these parameters are used in STEP1p and in STEP3, to correct the distortion on the points found on the calibration and speckle images, respectively.

This step is optional, and can be skipped if no distortion correction is required. If you are not sure whether you need distortion correction, you can run this step and assess the distortion parameters. If they are adequately small, you can choose to skip the distortion correction in STEP1p and STEP3. Alternatively, you can first assess the reprojection errors obtained in STEP1p without distortion correction, and if they are sufficiently small, STEP0 can be skipped.

Run STEP0_CalcDistortionParameters

1. Select whether this is a new analysis or a repeated one. Repeated analysis means you already ran the analysis for these cameras with the same images, but you want to repeat with a different distortion model. This option is faster than running a new analysis, because the checkerboard corner points need not by detected again.
2. Select one or multiple folders containing checkerboard images (one folder per camera).
3. Select whether or not to save the results. If 'Yes', select a folder for the results to be saved in.
4. Select whether or not to save the undistorted images.
5. Select the checkerboard parameters: number of rows, number of columns, and square size. Warning: the function detectCheckerboardPoints might cause this code to be slow if the number of images is very large.
6. Select the distortion model: number of coefficients for radial distortion (2 or 3), tangential distortion (0 or 1) and skew (0 or 1). Refer to [3] for details.

Step 0 script outputs

1. For each camera, a MATLAB structure cameraCBparameters is saved under the file name cameraCBparameters_cam_#, where # is the camera number taken from the folder name.
2. cameraCBparametersAllCams: a *Ncam*-by-1 cell array, where *Ncam* is the number of cameras, and each cell contains a cameraCBparameters structure.
3. The undistorted images, saved in a folder named "undistorted" in the same folder where the images are stored.
4. Figures plotting the calibration results:
   - For each camera, a figure showing the intrinsic and extrinsic parameters, and the reprojection errors. The first figure plots the results before distortion correction and the second plots the results after distortion correction. See for example Figure 7.

   - For each camera, a figure showing the reprojected points and the reprojection error statistics on each image (one figure for the original images and one figure after distortion correction). See for example Figure 8.

- For all cameras together, a figure showing the statistics of the camera intrinsic parameters before and after distortion correction.

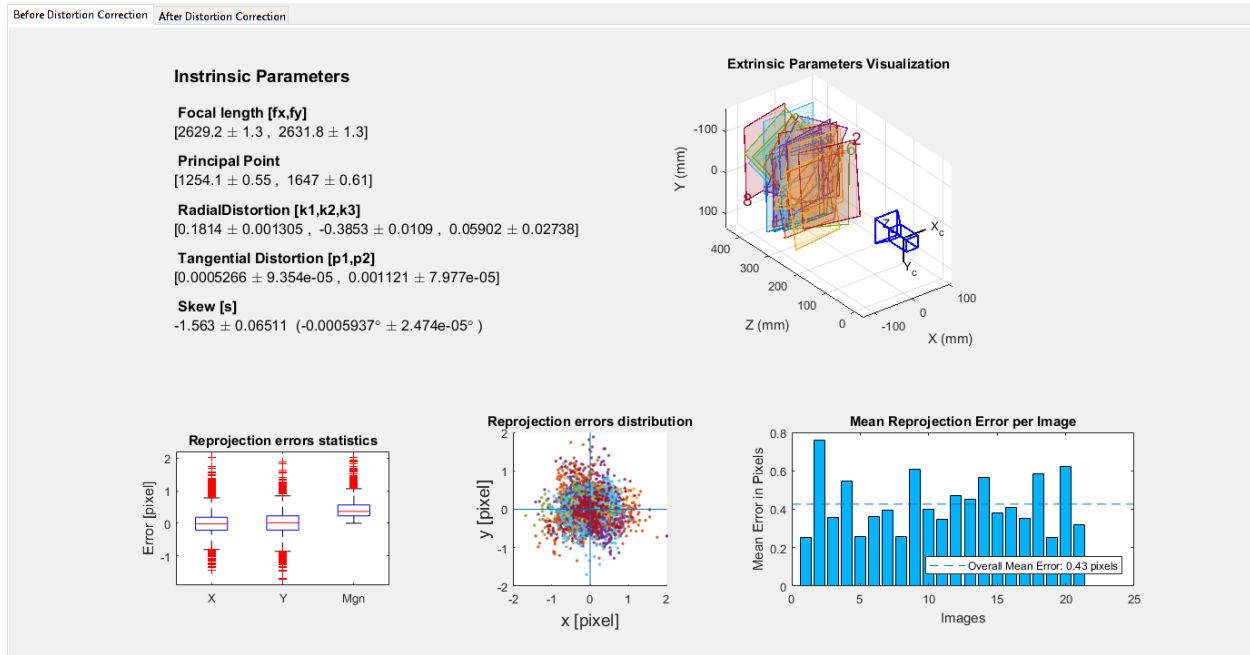- If the saving option was selected, a folder is created where all the results figures are saved.

Figure 7. Example of the results figure obtained in step 0 for one camera. The figure plots the camera's intrinsic parameters, a visualization fo the extrinsic parameters, and the reprojection error distribution of all points and all images, as well as the mean for each image and the statistics over all images.
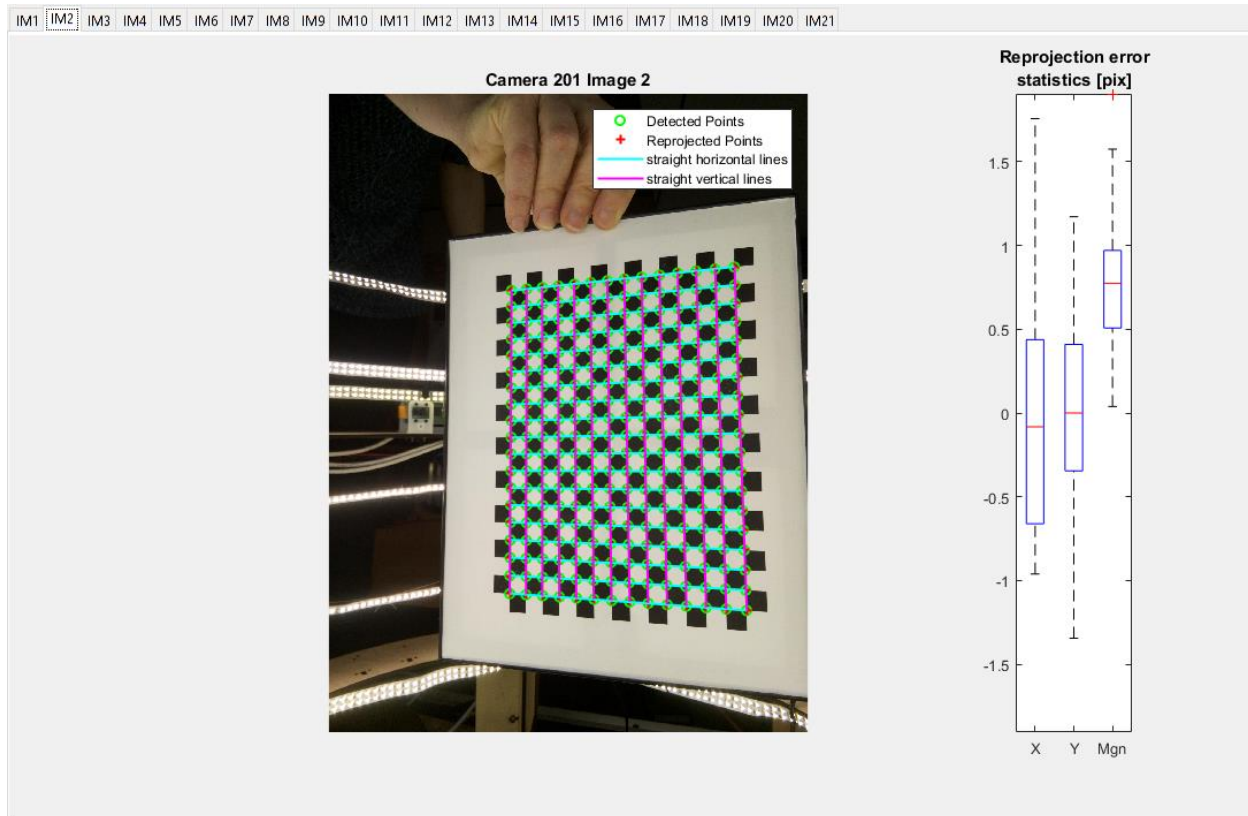
Figure 8. An example of the plot showing the detected and reprojected image points and the associated reprojected errors. Scroll through the tabs to view all images.

4.4   Step 1: Calculate DLT Parameters (Stereo Calibration)

`STEP1_CalcDLTparameters` is the main script for running as stereo calibration. The script utilizes the DLT method [9], whereby the closed-form solution of the mapping between 2D image points and 3D world points based on a distortion-free pin-hole camera model is obtained. It requires images of a non-planar calibration object with control points whose 3D positions in a global reference system are known with sufficient accuracy.

Run STEP1_CalcDLTparameters

1. Select the images of the 3D calibration target for the current analysis.
2. Select whether or not to save the results. If 'Yes', select a folder for the results to be saved in.
3. Select the MAT file containing the true 3D world coordinates of the calibration object.
4. The script will loop over all the selected cameras, and will perform the following steps for each camera:
     A. Turn the image to grayscale (if it is RGB).

     B. Open a GUI for masking the image, where you need to draw a polygon around the region of interest, which comprises of the portion of the image containing the calibration control points (Figure 9A). Use the zoom button in the figure, if necessary. The polygon is drawn by clicking on image points. At the end it is necessary to close the polygon by clicking again on the first point. After the polygon is closed, it is possible to move its vertices by dragging them (the cursor will turn to a circle). When the polygon is correctly completed, double click on it to finish.

C. The masked image will be displayed (B), and then the user has to approve it, or select to start over the masking GUI. When the mask is approved, the user has to enter the column numbers of the first and last columns inside the region of interest.

D. Based on the mask and the column numbers entered, the number of dots is calculated and the positions of the centroids of the black dots are calculated and plotted on the image. The user can modify the gray intensity threshold, by which the black dots are identified, to achieve more accurate centroid positions (Figure 9C). A higher threshold will increase the size of the regions and carry the risk of them merging together, and a lower threshold might cause regions to shrink and split or disappear. Usually, if the lighting conditions and the contrast are adequate on the entire image, the initial guess should be good. If different regions of the image have significantly different lighting (brightness), there won't be one threshold that fits the entire image. This means that you have to improve your experimental setup to have a more uniform lighting condition. Alternatively, modify the images using Photoshop or a similar tool (less recommended).

E. After selecting the threshold, the centroids are fixed, and are then sorted by columns counterclockwise (left to right) and by rows from bottom to top. The sorted points will be displayed on the image, and then the same procedure will start for the next camera (Figure 9D).

Step 1 script outputs

For each camera, a MAT file named `DLTstruct_cam_#` is saved, which contains the DLT parameters of camera number #, as well as the control points and column numbers used for calculation. These files can be then used to reconstruct the 3D positions of corresponding image points from stereo camera pairs (in STEP1p these are the calibration object control points and in in STEP3 these are the speckle image points).
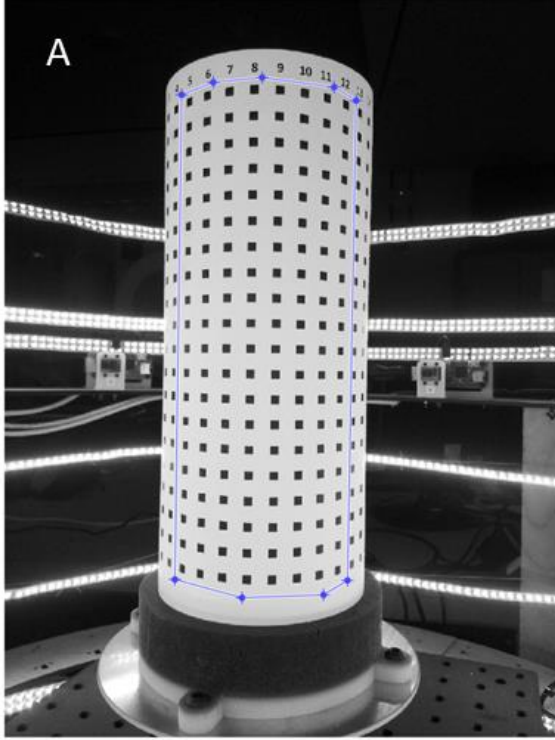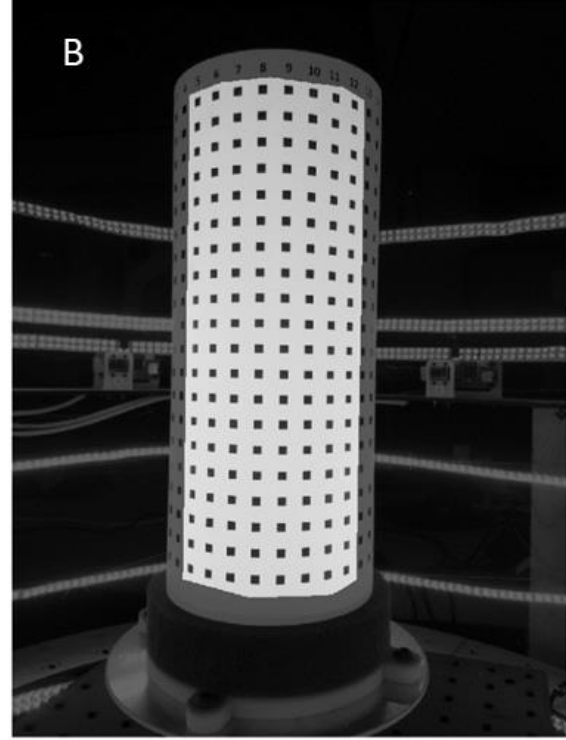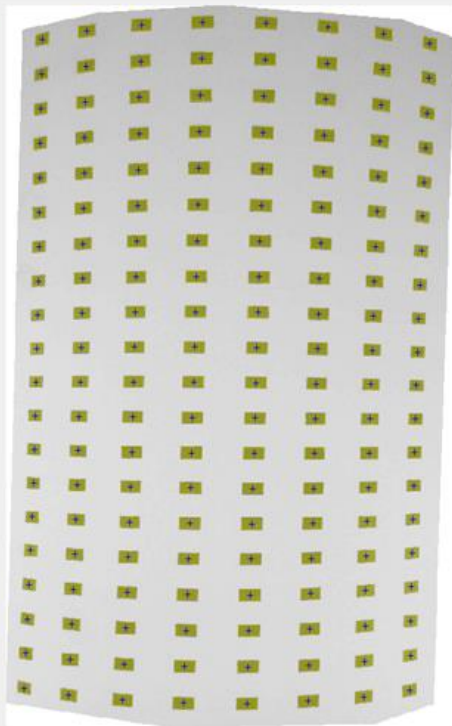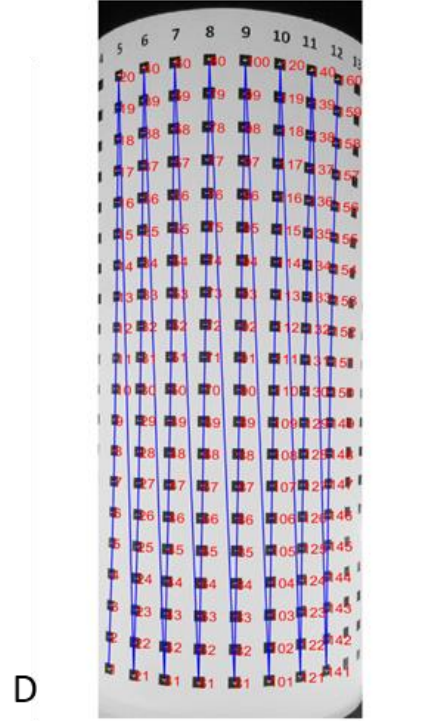
Figure 9. The steps of the DLT calibration process. (A) Draw a polygon around the calibration object points. (B) The image is masked based on the polygon. (C) The centroids are detected based on the detected gray level threshold between the white and black regions. Here you can change the threshold using the slide bar. (D) The centroids are sorted by rows and columns.

4.4.1 Step 1p: Calculate reconstruction errors

In this step, which is optional and not required for completing the analysis, the results from STEP1 are used to reconstruct the 3D positions of the calibration object control points from corresponding image points of camera pairs. Then, the reconstruction errors are calculated and plotted. This step is used to evaluate the reconstruction errors of the calibration step. Here, different distortion models can be used for distortion correction.

Run STEP1p_DLTreconstruction

1. Select DLTstruct files obtained in STEP1 from at least 1 camera pair (at least 2 cameras).
2. Select the indices of the camera pairs (pairs of cameras which have an overlapping region in their field of views). The format is 2 cameras in each row, e.g. if there are three pairs: (2,3) (3,4), (6,7), then type in [2 3; 3 6; 6 7].
3. Select whether or not to correct for distortion. If 'Yes', then select the distortion parameters calculated in STEP0 for the selected cameras.

Outputs of step 1p

1. The 3D reconstructed points and the corresponding reconstruction errors are calculated, and saved in a file named DLTstructPairs.
2. The reconstruction points and reconstruction errors are plotted for all cameras, and the figures are saved in the results folder. See Figure 10 for example.
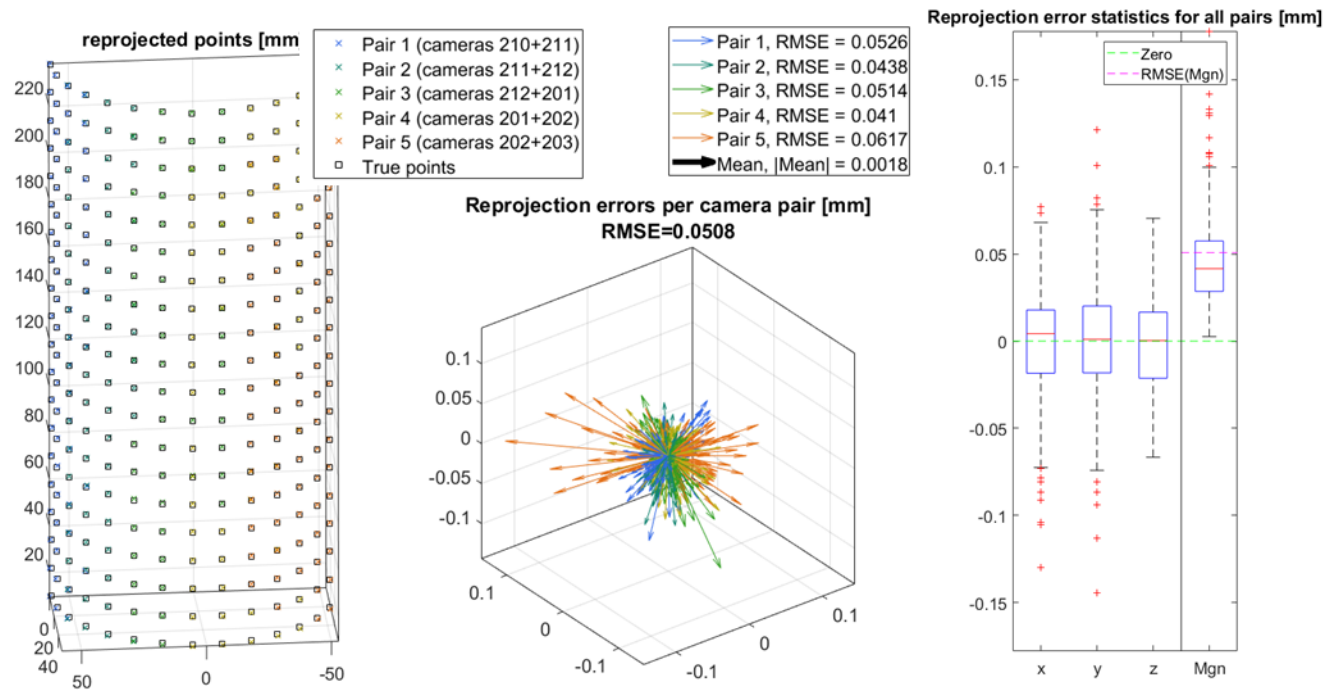
Figure 10. Example of results plot obtained in step 1p. The 3D points of the calibration object were reconstructed using images from 6 cameras which comprise 5 stereo-pairs. The reconstructed points are plotted against the true points (left), and the 3D reconstruction errors are plotted for each pair (center), and their statistics are reported as boxplots (right).

16

4.5 Step 2: 2D-DIC Using Ncorr

STEP2_2DDICusingNcorr is the main script for analyzing stereo images of the speckled object using 2D-DIC. This script utilized the open-source software Ncorr [10]. It can be performed either before or after STEP1. This step has to be performed once for each camera pair, and must be completed before the 3D points and surfaces can be reconstructed in Step 3. For each camera pair, one camera is considered as the *reference* camera and the other one is considered as the *deformed* camera. It means that images taken by the *deformed* camera are analyzed as deformed versions of the images taken by the *reference* camera.

Note that in this instruction manual, the same object was used both as the calibration object and as the speckled specimen. However, this is NOT a requirement. In fact, it is almost never the case. Your specimen and calibration object will likely be two different objects.

Run STEP2_2DDICusingNcorr

1. Select the folders of the *reference* camera and of the *deformed* camera, containing the speckle images.

2. Select saving and overwriting options.

3. A figure showing the image pairs from both cameras will appear (see Figure 11A). Review the image sets by clicking on the *play* button, or by scrolling using the arrows or the bottom bar. This figure is helpful for selecting the region of interest (ROI). If you clicked the *play* button, click the *stop* button and then click *enter* in MATLAB command window to continue to the Ncorr analysis.

4. Select a Region of Interest (ROI) option:

    o *New* means drawing a new mask using polygons. Select the number of ROIs. An ROI is defined by drawing a polygon on the reference image (see Figure 11B). Make sure that the ROI is visible on all the images (from both cameras). Click on the image to select the first vertex and continue placing new vertices by clicking on image points. To close the polygon, click on the first vertex. Once the polygon is closed, vertices can be moved by dragging them (the cursor will turn to a circle when you hover over a vertex). Also, the entire can be translated by dragging. To finish, double-click on the polygon.

    o *Saved* means you already have a saved mask and you want to use it again. For example, if you ran the analysis and now you just want to run it again with different options in Ncorr such as subset size or spacing, etc. In this case, you need to select an ROI for the correct camera pair.

    o *Ncorr* means you will draw the ROI in Ncorr. Ncorr has more options for drawing shapes and cutting holes in the ROI, but the GUI is smaller and you don't have the option to scroll through the images to assist with ROI placement.
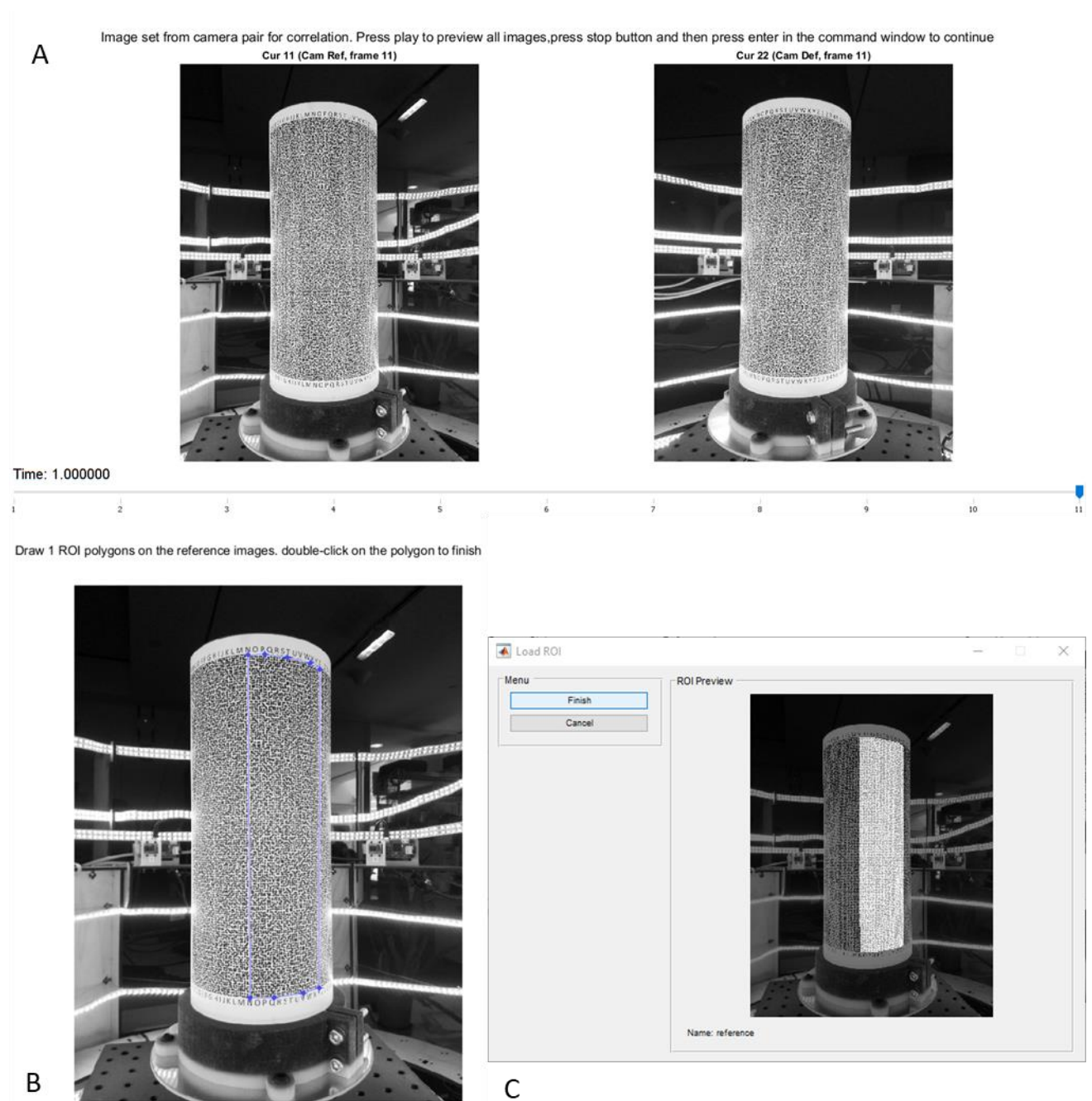
Figure 11. (A) Review the stereo image set by clicking *play* or scrolling. (B) Select the ROI by drawing a polygon (if the *New* ROI option selected. (C) The ROI opened in Ncorr.

5. Next, the Ncorr window will appear, where the 2D-DIC analysis is performed. The following steps have to be completed in this window:

    A. If you already set the ROI, the ROI will be displayed (see Figure 11C). Click Finish. If not, draw an ROI in Ncorr (refer to Ncorr instruction manual for details if necessary).

B. When the ROI is set, the next step is to set the DIC parameters. Select the *Analysis* tab on the top menu, then *Set DIC parameters*. A window will pop up where the *Subset radius* and *Subset spacing* can be selected (see Figure 12). These parameters are visualized on the image on the right. These are some points to consider when selecting these parameters:

- The subset radius should be large enough to include at least 2-3 speckles along its width and height, in order to contain enough unique and identifiable features.

- The subset radius should be small enough to satisfy the assumption that the deformation is homogeneous inside the subset.

- The subset spacing determines the distance between data points. Small spacing means the point grid will be denser. As a result, the analysis will take longer to run, but the resolution of the 3D reconstructed surface will be higher. It is recommended to select a subset spacing equal to half of the subset radius.

- More information on subset size and spacing selection can be found in [1], [11].

C. Select the desired parameters and click Finish. There is no need to change the other options. A window showing all the selected parameters will pop up. Click Yes.
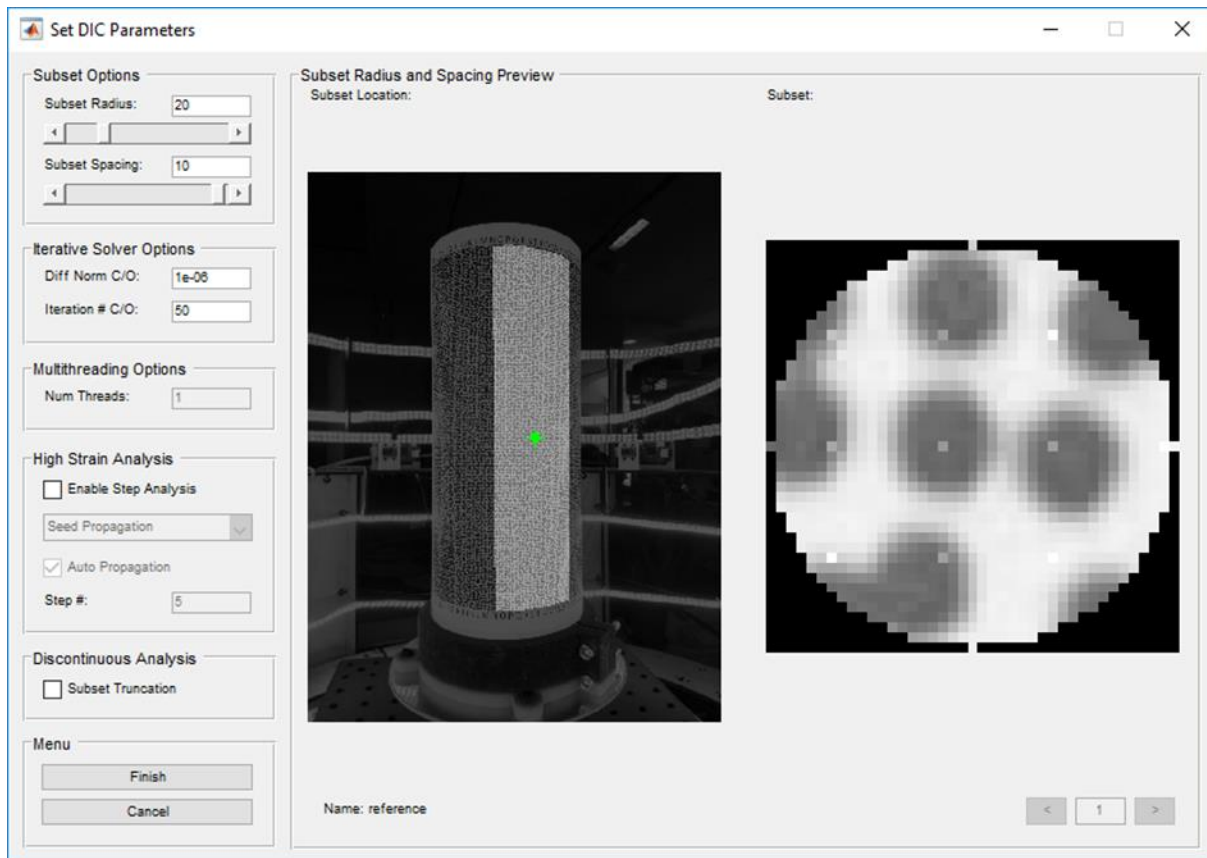
Figure 12. Subset radius and Subset spacing selection in Ncorr. Select a subset radius such that the circle contains at least 2-3 well defined speckles. The green point can be dragged over the ROI to inspect speckles in different regions.

D. Select the *Analysis* tab, then *Perform DIC analysis*. A *Select region* window will pop up. Click on *Select region* and then on one of the white regions on the image (if the ROI is composed of only one region, then you will have only one option).

E. After selecting a region, a *Set seeds* window will pop up. Click on *Set seeds* and then click on a point inside the ROI. Select a point that is clearly visible from both views, and that is not too close to the edges of the ROI. See Ncorr documentation for more details on optimal seed placement, if necessary. Click Finish.

F. When *processing seeds* is finished, a *Seed Preview* window will pop up (see Figure 13). Scroll through all the images to make sure that the seed point was detected correctly in all of them. If they look correct, click *Finish*. If the correlation coefficient between the subsets around the detected seeds is too high, an error prompt will appear. This usually means that the seed point was not detected correctly on at least one of the images. If this is the case, click *cancel*, move the seed to a better position, and try again until seed placement is successful.
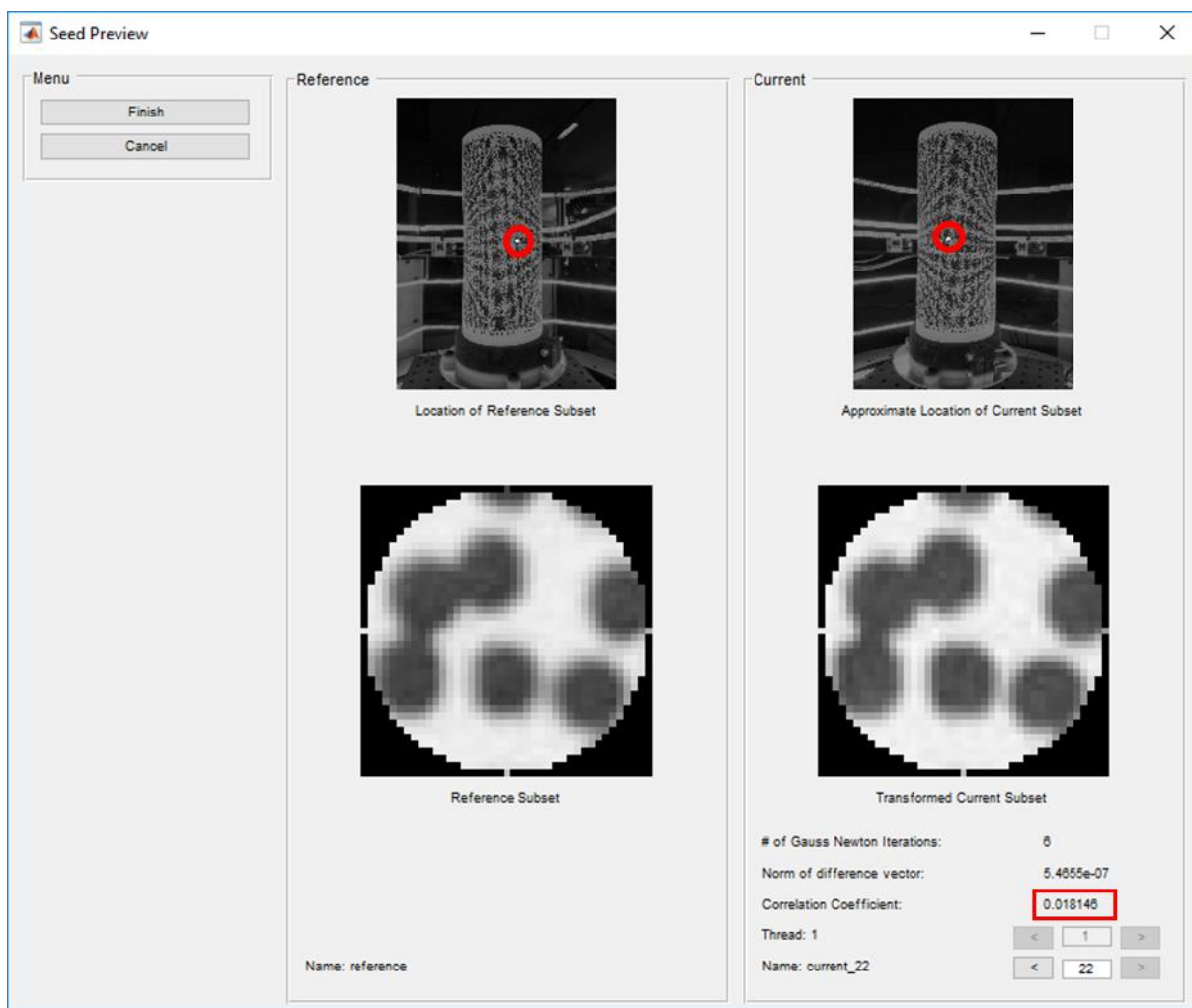
Figure 13. The seed preview window. Scroll through all current images, and ensure that the seeds were placed correctly by inspecting their positions on the images (here marked in red circles), and the value of the correlation coefficient (here in red rectangle). If the seed point is placed in a region which is very deformed due to the angled view, the seed placement might occasionally be wrong. In this case, place the seed in a better position and try again.

G. When the seeds are properly placed and you click *Finish*, the DIC analysis will start running, propagating from the seed points to the rest of the ROIs. When the analysis is done, a message will pop up stating that *DIC analysis completed successfully*. Press *OK* to finish.

H. Click on *Analysis* once again, and select *Format displacements*. There is no need to select anything in this window or change the settings. Just click *Finish* and then *Yes*. Since this is a 2D analysis of stereo images, the displacements here do not have a physical meaning that you can easily inspect. Only after the 3D reconstruction step, you will be able to see the 3D displacements.

I. At this point the Ncorr analysis is complete. There is no need to run the *Calculate strains* part. Go back to MATLAB main window **without closing the Ncorr window yet** and press enter in the command window. Then, the results will be imported from Ncorr, and a results structure named `DIC2DpairResults_C_#1_C_#2` will be saved, where #1 is the index of the *reference camera* and #2 is the index of the *deformed camera.*

J. You might get a warning from Ncorr: *prior DIC has been detected and will be deleted*. You can click *Yes*, as all the necessary analysis results for 3D-DIC are saved outside Ncorr.

6. Select if you want to plot the results now. If you select *Yes*, you will be requested to select if you want to change the limits of the correlation coefficients for display (leave this blank to use the default limits, which are between 0 and the maximum value of the correlation coefficient found in this analysis. Three animation figures will appear:

   • Figure 1 plotting the reference image on the left and all the current images on the right (from both views). Corresponded points are displayed on the images with the color depicting the value of the correlation coefficient (see Figure 14).

   • Figure 2 plotting the same as 1, but the images from the two views are plotted on the left and right subplots separately.

   • Figure 3 plotting the same as 3, but instead of points, the triangular faces are plotted, and the face colors represent the combined (maximal) correlation coefficient of the three vertices.

Note: You can also run the function `plotNcorrPairResults` separately, after the results are stored. If you run `plotNcorrPairResults` without any input, you will be requested to select a `DIC2DpairResults` structure by browsing. Otherwise, if the `DIC2DpairResults` is already in the workspace, you can give it as an input to the function: `plotNcorrPairResults(DIC2DpairResults)`.
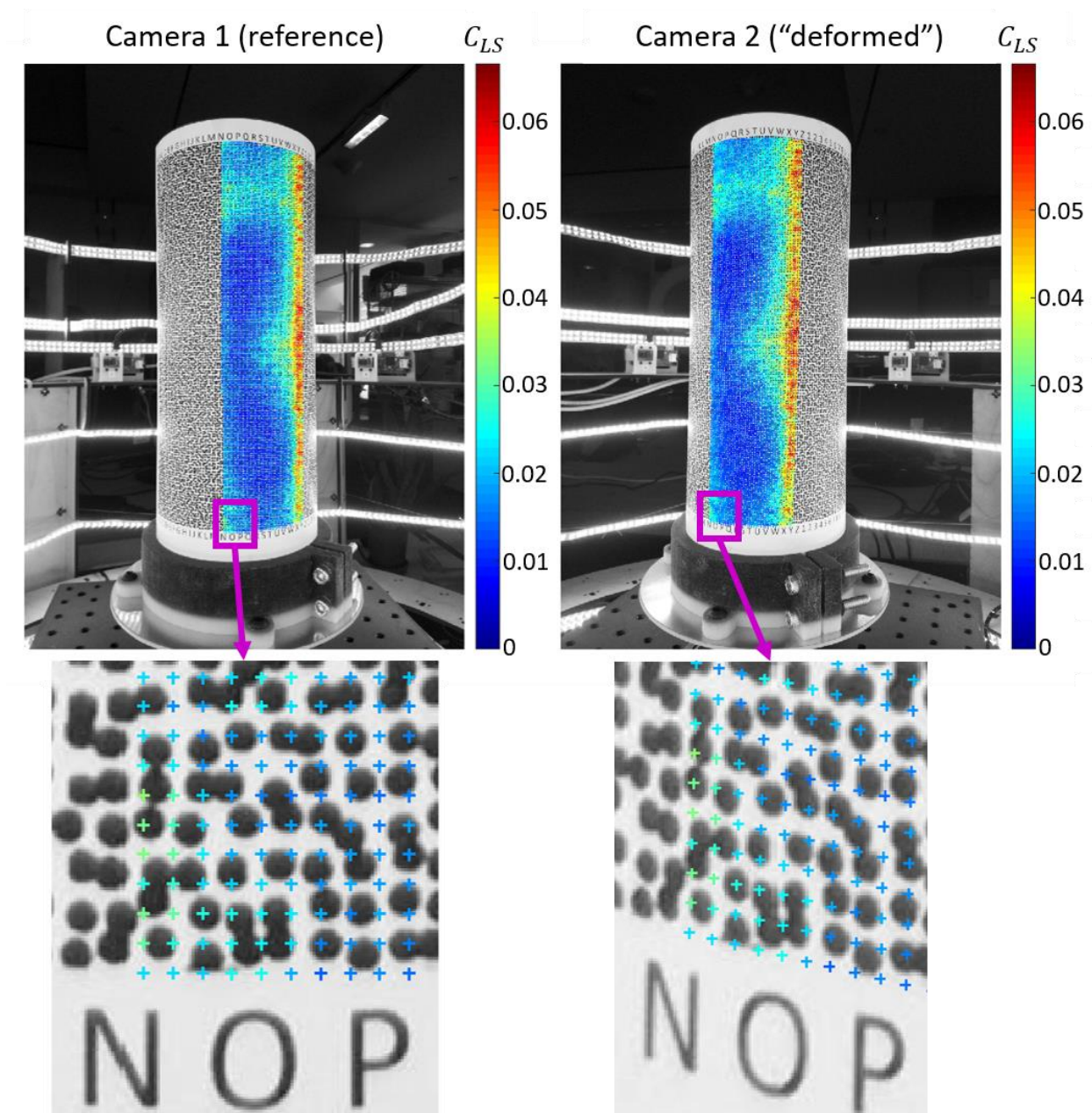
Figure 14. 2D-DIC results on a set of stereo images. Corresponding points are plotted with colors depicting the matching correlation coefficients. Higher correlation coefficient are obtained in regions where the deformation due to the stereo angle is higher.

4.6 Step 3: 3D reconstruction

STEP3_3Dreconstruction is the main script for transforming pairs of corresponding points from the speckle images obtained in step 2 into 3D points and surfaces using the DLT parameters obtained in step 1. For each camera pair, a dynamic 3D surface will be reconstructed, the multiple surfaces can be stitched together, and the combined surface saved and plotted. Here, you can also select distortion correction options. If distortion correction is selected,

the speckle image points will be corrected as well as the DLT parameters, based on the distortion parameters selected.

Run STEP3_3Dreconstruction

1. Select the following:

   A. One or more 2D-DIC results files (`DIC2DpairResults`) to be reconstructed.

   B. The folder where the DLT parameters (`DLTstruct_cam_#`) for the relevant cameras are stored.

   C. Distortion correction options. If distortion correction is required, you will be asked to select the `cameraCBparameters_cam_#` of the relevant cameras.

   D. Saving options.

2. The following steps are then performed for each camera pair:

   A. If distortion correction is required, the 2D points and the DLT parameters are corrected according to the distortion parameters of each camera. Warning: distortion correction might be time consuming if the number of points is large.

   B. The 3D points are reconstructed using the DLT algorithm.

3. Select whether or not to stitch the surfaces together (Warning: the stitching procedure is pretty slow at the moment. Its speed will be improved in next releases). If you choose to stitch, you will be asked to select the indices of the surfaces you want to stitch together. The stitching algorithm will display the surfaces being stitched one after the other. An example of the stitching process is given in Figure 15.

4. When stitching is complete, or if no stitching was required, the complete surface will be plotted as animation figure, and can be played to check its dynamic behavior.

5. The results of the 3D reconstruction are saved in a structure named `DIC3Dcombined_#PairsResults`, where # is the number of camera pairs in this analysis.
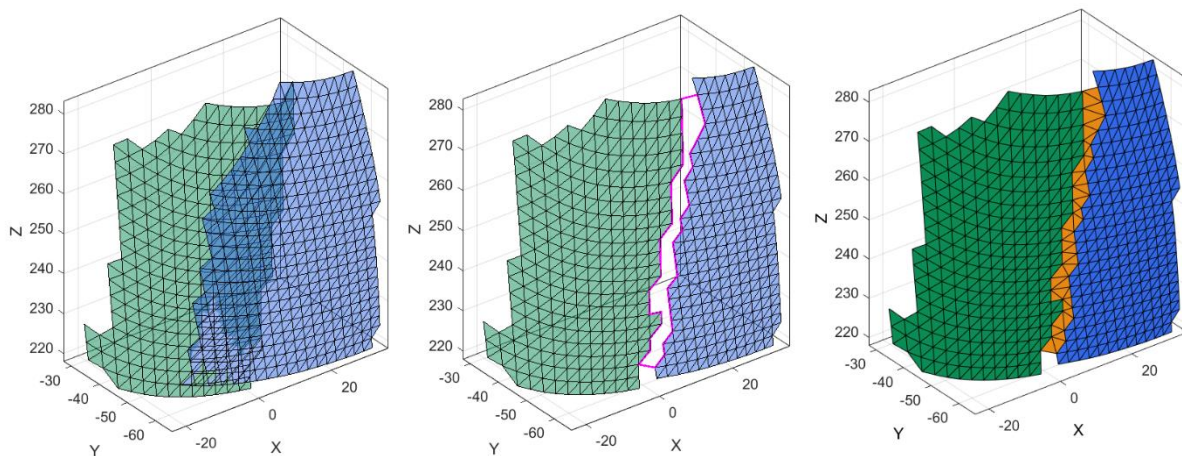
Figure 15. Stitching of overlapping surfaces obtained from two camera-pairs. First, the overlapping regions is removed, then the gap is stitched. The stitching algorithm preserves the original vertices, and does not add new vertices of modify the positions of existing vertices.

4.7   Step 4: Post processing

`STEP4_PostProcessing` is the main script for post-processing of the 3D surfaces obtained in step 3. Based on the dynamic positions of the 3D reconstructed points, the displacement and rigid body motion are calculated in each configuration (each time step). In addition, for each triangular face, the deformation and strain measures are calculated and plotted, and all the results are saved as a MATLAB structure.

Run STEP4_PostProcessing

1. Select the `DIC3Dcombined_#PairsResults` file you saved in STEP3.
2. Select saving options.
3. The displacements, rigid body motion, deformation, and strains, will be calculated.
4. Select whether or not you want to plot the results now. If you select *Yes*, you will be requested to select which measures you want to plot (see Figure 16). Click "Select All" to plot all the parameters. Click "Select to remove rigid body motion" if you want to visualize the dynamic shape and associated parameters with the rigid body motion subtracted from the positions of the vertices.
5. Each measure you selected will be plotted in a separate animation figure, where the static/dynamic behavior of the measure can be viewed (see Figure 17 for a few examples).
6. The results of the post-processing are saved in a structure named `DIC3DPPresults_#PairsResults`, where # is the number of camera pairs in this analysis.

Note: You can also run the function `plot3DDICPPresults` independently, after the results of step4 are stored. If you run the function `plot3DDICPPresults` by typing it in the command window without any input, you will be requested to select a `DIC3DPPresults_#PairsResults` result structure by browsing. Otherwise, if `DIC3DAllPairsResults` is already in the MATLAB workspace, you can give it as an input to the function, like this: `plot3DDICPPresults(DIC3DPPresults);`
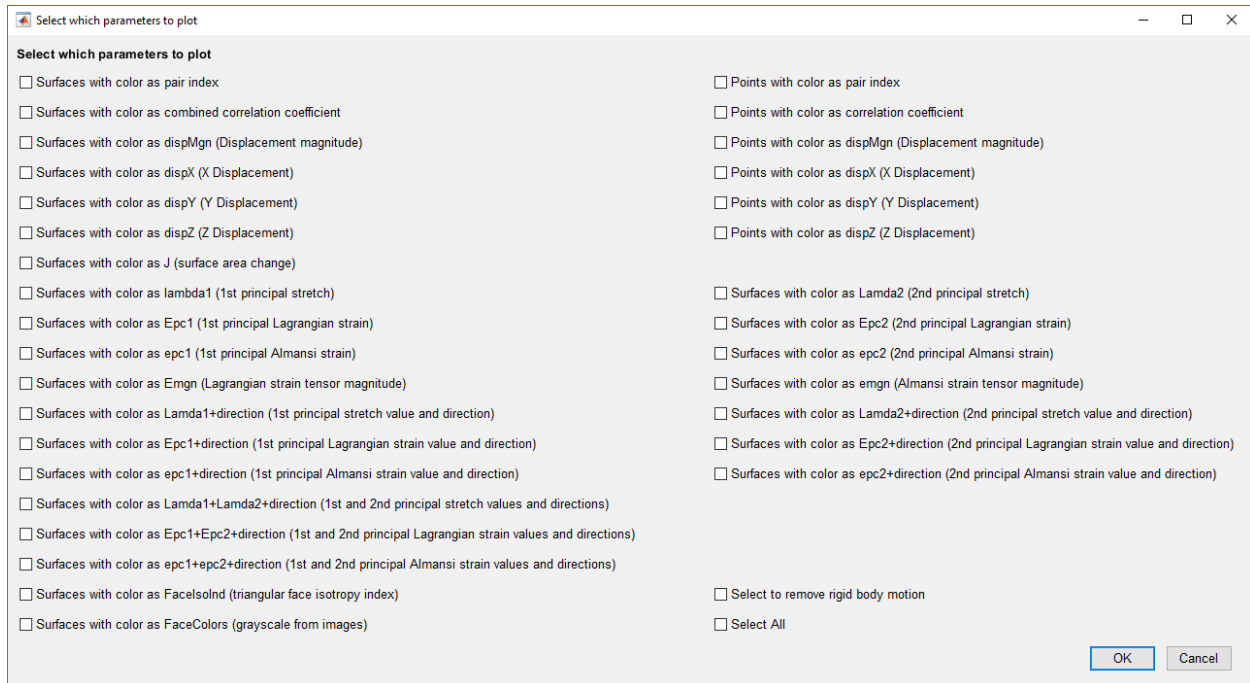
Figure 16. Select plot options.
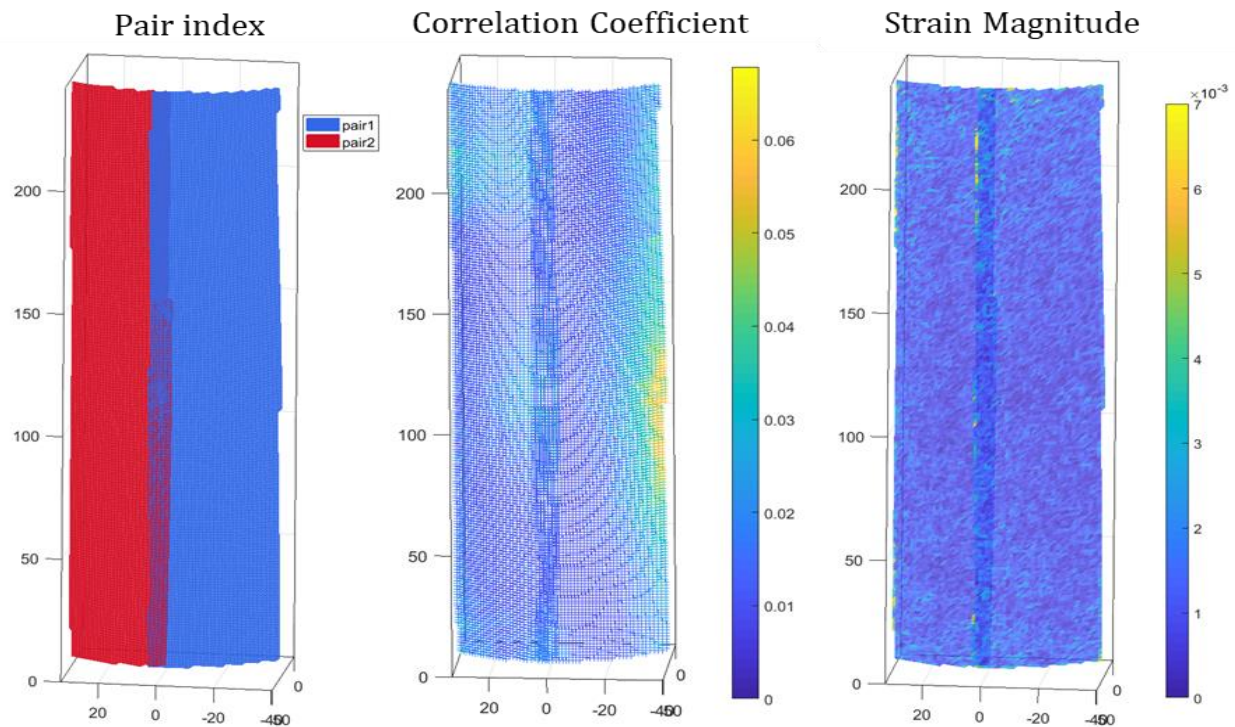
Figure 17. Examples of a few plotting options for 3D-DIC results. The figure shows two reconstructed surfaces, reconstructed from three cameras (two adjacent camera pairs). The triangular faces are plotted with the colors depicting the index of the pair (left). The vertices are plotted with the colors depicting the combined correlation coefficient (center). The faces are plotted with the colors depicting the strain magnitude (right). In this example, since the object was rigidly moved, the strain represents the measurement error of the raw data (without smoothing).

5. A Deeper Look into MultiDIC

This section provides a more detailed description of the codes, and is targeted for users who wish to get a deeper understanding of it, to modify the codes, or to add new functions or batch scripts.

5.1 Step 0

- The initial parameters that appear in the dialog boxes (e.g. checkerboard parameters and distortion model) can be changed in the code to appear as the default options.

- Each of the fields of the `cameraCBparameters` structure is described in the following table:

Field	Sub-field	Class/size	Description
`icam`		integer	The index of the camera (as given by the user in the name of the folder).
`imagesUsed`		1-by-N array integer	A vector containing the indices of the images used for calibration (images might be discarded if the checkerboard points cannot be detected correctly).
`boardSize`		1-by-2 array integer	Checkerboard dimensions, as a 2-element [*height*, *width*] vector. The dimensions of the checkerboard are expressed in terms of the number of squares. *Height* must be uneven and *width* must be even.
`squareSize`		double	Checkerboard square size in [m] units.
`imagesInfo`		Struct	
	`Nimages`	integer	The number of checkerboard images selected for calibration.
	`imageFileNames`	*1*-by-*Nimages* cell array of chars	The paths for each of the images.
	`imageType`	char	The image file format, indicating the standard file extension, such as 'jpg','png','tif', etc..
	`imageSize`	*1*-by-*2* integer	Image size in pixels, specified as [*rows*, *columns*] or [*height, width*].
`imagePoints`		Npoints-by-2-by-Nimages double	The detected checkerboard corner coordinates for all images. The second dimension refers to the [x,y] coordinates
`cameraParameters`		Struct	a structure containing the standard MATLAB `cameraParameters` object (https://www.mathworks.com/help/vision/ref/cameraparameters.html)
	`RadialDistortion`	1-by-3 or 1-by-2 double	The radial distortion coefficients [*k1, k2*] or [*k1, k2, k3*], according to the selected camera model.
	`TangentialDistortion`	1-by-2 double	The tangential distortion coefficients [*p1, p2*].
	`Skew`	double	The camera axes skew, specified as a scalar representing the angle between the axes.

	NumRadialDistortionCoefficients	2 or 3 (double)	Number of radial distortion coefficients
	EstimateSkew	logical	Estimate skew flag. When set to *true*, the object estimates the image axes skew. When set to *false*, the image axes are estimated to be exactly perpendicular.
	EstimateTangentialDistortion	logical	Estimate tangential distortion flag. When set to *true*, the tangential distortion parameters are estimated. When set to *false*, it is assumed that the tangential distortion is negligible and `TangentialDistortion` is set to [0,0].
	PrincipalPoint	1-by-2 double	The optical center [*cx,cy*] in pixels, representing the coordinates of the optical center of the camera.
	FocalLength	1-by-2 double	The focal length in the *x* and *y* directions [*fx*, *fy*] in pixel units, where: *fx*=F**sx and *fy*=F**sy. F is the focal length in world units, typically in millimeters, and [*sx*, *sy*] are the number of pixels per world unit.
	WorldPoints	Npoints-by-2 double	World coordinates of the corner points on the checkerboard pattern.
	WorldUnits	char	World points units, specified as a character vector (typically 'mm').
	Numpatterns	integer	Number of calibration patterns (number checkerboard images) that estimates camera parameters.
	ReprojectedPoints	Npoints-by-2-by-Nimages double	World points reprojected onto the calibration images.
	ReprojectionErrors	Npoints-by-2-by-Nimages double	The difference between the detected and reprojected points.
	MeanReprojectionError	double	The average Euclidean distance between reprojected and detected points, specified in pixels.
estimationErrors			The standard errors structure of estimated camera parameters, returned as a MATLAB cameraCalibrationErrors object, (https://www.mathworks.com/help/vision/ref/cameracalibrationerrors.html). The estimation errors represent the uncertainty of each estimated parameter (the standard error corresponding to each estimated camera parameter). The returned standard error σ (in the same units as the corresponding parameter) can be used to calculate confidence intervals. For example $\pm1.96\sigma$ corresponds to the 95% confidence interval. In other words, the probability that the actual value of a given parameter is within 1.96σ of its estimate is 95%.

			Same as `cameraParameters`, but after the distortion correction (AUD stands for After UnDistortion).
`cameraParametersAUD`			
`estimationErrorsAUD`			Same as `estimationErrors`, but after the distortion correction (AUD stands for After UnDistortion).
`imagePointsAUD`			Same as `imagePoints`, but after the distortion correction (AUD stands for After UnDistortion)..

5.2 Step 1

- Each of the fields of the `DLTstructCam` structure is described in the following table:

Field	Sub-field	Class/size	Description
`indCam`		integer	camera index
`DLTparams`		11-by-1 double	The 11 DLT parameters
`columns`		1-by-Ncolumns	The indices of the columns of the calibration object
`imageCentroids`		Npoints-by-2 double	The centroids image coordinates (pixels)
`C3Dtrue`		Nrows-by-Ncolumns-by-3 double	True 3D coordinates of the calibration object arranged by rows (bottom to top) - columns (left to right) -xyz

5.3 Step 1p

- Each of the fields of the `DLTstructPairs` structure is described in the following table:

Field	Sub-field	Class/size	Description
`indCams`		1-by-Ncams integer	Array of the camera indices
`indPairs`		Npairs-by-2 integer	Array of the camera indices of each camera stereo pair. Each row represents a pair, where the first column is the reference camera and the second column is the deformed column.
`DLTparams`		1-by-Ncams cell array	In each cell, the DLT parameters 11-by-1 double array.
`columns`		1-by-Ncams cell array	the indices of the columns of the calibration object used by each camera
`imageCentroids`		1-by-Ncams cell array	in each cell the Npoints -by-2 array of the centroids image coordinates
`truePoints`		Nrows-by-Ncolumns-by-3 double	True 3D coordinates of the calibration object arranged by rows (bottom to top) - columns (left to right) -xyz
`reconstructPoints`		1-by-Npairs cell array	In each cell, an Npoints-by-3 array of the 3D coordinates of the reconstructed points seen by the pair of cameras.

Field	Sub-field	Class/size	Description
reconstructErrors		1-by-Npairs cell array	In each cell, an Npoints-by-3 array of the 3D errors (difference vector) between the reconstructed points and the true points
distortionModel		1-by-Ncams cell array	In each cell, the intrinsic camera parameters (cameraParameters object) used for correcting the distortion on the image points and associated DLT parameters. If no distortion correction was performed, it is set to 'none'.

5.4 Step 2

- Each of the fields of the DIC2DpairResults structure is described in the following table:

Field	Sub-field	Class/size	Description
nCamRef		integer	index of the reference camera of the pair
nCamDef		integer	index of the deformed camera of the pair
nImages		integer	number of images in the set (time frames, including the reference)
IMpaths		2*nImages-by-1 cell array	Each cell contains the paths to one images
ROImask		imageSize1-by-imageSize2 logical	Matrix the same size as the reference image, with 1 in the pixels inside the ROI and 0 outside the ROI.
Points		2*nImages-by-1 cell array	Each cell contains an Npoints-by-2 array of the correlated points between this image and the reference image (the points indices are corresponding between all images. Points can have NaN values if they could not be matched in some images).
CorCoeffVec		2*nImages-by-1 cell array	Each cell contains an Npoints-by-1 array of the correlation coefficients between the current image and the reference image (the points indices are corresponding).
Faces		Nfaces-by-3 integer	Each row represents a triangular face, and the 3 columns represent the indices of the vertices of that triangular face.
FaceColors		Nfaces-by-1 Uint8	Each row represents a triangular face, and the value represent the grayscale intensity from the reference image, as a mean of the grayscale intensity of the three pixels where the three vertices of the triangles are located.
ncorrInfo		struct	handles_ncorr.data_dic.dispinfo containing ncorr information, such as the subset radius and subset spacing.

5.5 Step 3

- Each of the fields of the DIC3Dcombined structure is described in the following table:

Field	Sub-field	Class/size	Description

Field	Sub-field	Class/size	Description
pairIndices		nPairs-by-2 integer	In each row the camera indices [nRef nDef] for each pair
Faces		Nfaces-by-3 integer	The vertex indices of all the triangular faces
distortionModel		1-by-2 cell array	The distortion model (camera intrinsic parameters) used for each of the cameras in the pair.
FaceColors		Nf-by-1	The greyscale pixel intensities imposed on the faces (from the reference image)
Points3D		Nframes-by-1 cell array	Each cell represents a time frame and contains Npoints-by-3 array of the 3D coordinates of all the points.
corrComb		Nframes-by-1 cell array	Each cell contains an Npoints-by-1 array corresponding to the combined correlation coefficient of each 3D point.
FacePairInds		Nf-by-1 integer	For each face, a number representing the pair index. These cameras for pair index can be retrieved from the field pairIndices.

5.6   Step 4

- Each of the fields of the DIC3Dcombined structure is described in the following table:

Field	Sub-field	Class/size	Description
pairIndices		nPairs-by-2 integer	In each row the camera indices [nRef nDef] for each pair
Faces		Nfaces-by-3 integer	The vertex indices of all the triangular faces
FaceColors		Nf-by-1	The greyscale pixel intensities imposed on the faces (from the reference image)
RBM	RotMat	Nframes-by-1 cell array	Each cell represents a time frame and contains 3-by-3 rotation matrix.
	TransVec	Nframes-by-1 cell array	Each cell represents a time frame and contains 3-by-1 translation vector.
Points3D		Nframes-by-1 cell array	Each cell represents a time frame and contains Npoints-by-3 array of the 3D coordinates of all the points.
Points3D_ARBM		Nframes-by-1 cell array	Each cell represents a time frame and contains Npoints-by-3 array of the 3D coordinates of all the points, after the rigid body motion has been subtracted from it.
corrComb		Nframes-by-1 cell array	Each cell contains an Npoints-by-1 array corresponding to the combined correlation coefficient of each 3D point.
FacePairInds		Nf-by-1 integer	For each face, a number representing the pair index. These cameras for pair index can be retrieved from the field pairIndices.

FaceIsoInd		struct	Nf-by-1 array corresponding to the isotropy index of each face in the reference configuration.
Deform and Deform_A		structure	A structure containing all the deformation parameters, containing the fields below, each of the fields contains a is a Nframes-by-1 cell array (one cell for each time frame), so the specified size refers to what is inside each cell.
	Fmat	3-by-3-by-Nfaces double	Deformation gradient tensor
	J	Nfaces-by-1 double	Dilitation (det(Fmat))
	Cmat	3-by-3-by-Nfaces double	Right Cauchy-Green deformation tensor
	Lamda1	Nfaces-by-1 double	First principal surface stretch (smallest)
	Lamda2	Nfaces-by-1 double	Second principal surface stretch (largest)
	Emat	3-by-3-by-Nfaces double	Lagrangian strain tensor
	emat	3-by-3-by-Nfaces double	Almansi strain tensor
	Emgn	Nfaces-by-1 double	Lagrangian strain tensor magnitude
	emgn	Nfaces-by-1 double	Almansi strain tensor magnitude
	d3	Nfaces-by-3 double	Face normal in current configuration
	Epc1	Nfaces-by-1 double	First principal surface Lagrangian strain (smallest)
	Epc2	Nfaces-by-1 double	Second principal surface Lagrangian strain (largest)
	Epc1vec	Nfaces-by-3 double	First principal surface Lagrangian strain direction (largest)
	Epc1vecCur	Nfaces-by-3 double	First principal surface Lagrangian strain direction (largest), transformed into the current configuration
	Epc2vec	Nfaces-by-3 double	Second principal surface Lagrangian strain direction (smallest).
	Epc2vecCur	Nfaces-by-3 double	Second principal surface Lagrangian strain direction (smallest), transformed into the current configuration
	epc1	Nfaces-by-1 double	First principal surface Almansi strain (smallest)
	epc2	Nfaces-by-1 double	Second principal surface Almansi strain (largest)
	epc1vec	Nfaces-by-3 double	First principal surface Almansi strain direction (smallest).
	epc2vec	Nfaces-by-3 double	Second principal surface Almansi strain direction (largest).

	`EShearMax`	Nfaces-by-1 Double	Max Lagrangian shear strain
	`eShearMax`	Nfaces-by-1 Double	Max Almansi shear strain
	`EShearMaxVec1`	Nfaces-by-3 Double	Max Lagrangian shear strain direction 1 in the reference configuration
	`EShearMaxVec2`	Nfaces-by-3 Double	Max Lagrangian shear strain direction 2 in the reference configuration
	`EShearMaxVecCur1`	Nfaces-by-3 Double	Max Lagrangian shear strain direction 1 in the deformed configuration
	`EShearMaxVecCur2`	Nfaces-by-3 double	Max Lagrangian shear strain direction 1 in the deformed configuration
	`eShearMaxVec1`	Nfaces-by-3 double	Max Almansi shear strain direction 1 in the reference configuration
	`eShearMaxVec2`	Nfaces-by-3 double	Max Almansi shear strain direction 2 in the reference configuration
	`Eeq`	Nfaces-by-1 double	Equivalent (Von-Mizes) Lagrangian strain
	`eeq`	Nfaces-by-1 double	Equivalent (Von-Mizes) Almansi strain
	`Area`	Nfaces-by-1 double	Face area

References

[1] M. A. Sutton, "Digital Image Correlation for Shape and Deformation Measurements," *Springer Handb. Exp. Solid Mech.*, pp. 565–600, 2008.

[2] Y. L. Dong and B. Pan, "A Review of Speckle Pattern Fabrication and Assessment for Digital Image Correlation," *Exp. Mech.*, vol. 57, no. 8, pp. 1161–1181, Oct. 2017.

[3] "What Is Camera Calibration? - MATLAB & Simulink." [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html.

[4] Z. Zhang, "A Flexible New Technique for Camera Calibration (Technical Report)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, 2000.

[5] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1106–1112, 1997.

[6] J. Bouguet, "Camera Calibration Toolbox for Matlab." [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/. [Accessed: 25-Jan-2018].

[7] G. Bradski and A. Kaehler, *Learning OpenCV*, First Edition. O'Reilly Media, Inc., 2008.

[8] P. L. Reu, "A Study of the Influence of Calibration Uncertainty on the Global Uncertainty for Digital Image Correlation Using a Monte Carlo Approach," *Exp. Mech.*, vol. 53, no. 9, pp. 1661–1680, Nov. 2013.

[9] Y. I. Abdel-Aziz, "Direct linear transformation from comparator coordinates in close-range photogrammetry," *Proc. Am. Soc. Photogramm. Symp. close-range Photogramm. Falls Church (VA). Am. Soc. Photogramm. Symp. Close-Range Photogramm. 1971*, pp. 1–19, 1971.

[10] J. Blaber, B. Adair, and A. Antoniou, "Ncorr: Open-Source 2D Digital Image Correlation Matlab Software," *Exp. Mech.*, vol. 55, no. 6, pp. 1105–1122, 2015.

[11] B. Pan, H. Xie, Z. Wang, K. Qian, and Z. Wang, "Study on subset size selection in digital image correlation for speckle patterns," *Opt. Express*, vol. 16, no. 10, p. 7037, May 2008.