Augmented Lagrangian Digital Image Correlation (AL-DIC) Code Manual (v4.2)

Jin Yang[1,2†], Kaushik Bhattacharya[2‡]

[1] Department of Mechanical Engineering, University of Wisconsin-Madison
[2] Division of Engineering and Applied Science, California Institute of Technology

Email: [†]jyang526@wisc.edu; [‡]bhatta@caltech.edu

Github page: https://github.com/jyang526843/2D_ALDIC
MATLAB FileExchange page: https://www.mathworks.com/matlabcentral/fileexchange/70499-augmented-lagrangian-digital-image-correlation-and-tracking

Last time updated on June 15, 2021

Contents

1 Introduction

Digital image correlation (DIC) technique is a powerful experimental tool for measuring full-field displacements and strains. Most current DIC algorithms can be categorized into either *local* or finite-element-based *global* methods, see Fig. 1. As with most experimental approaches, there are drawbacks with each of these methods. In the local method the subset deformations are estimated independently and the computed displacement field may not necessarily be kinematically compatible. Thus, the deformation gradients can be noisy, especially when using small subsets. Although the global method often enforces kinematic compatibility, it generally incurs substantially greater computational costs than its local counterpart, which is especially significant for large data sets. Here we present a new hybrid DIC algorithm, called *augmented Lagrangian digital image correlation (ALDIC)* [1], which combines the advantages of both the local (fast computation times) and global (compatible displacement field) methods.

ALDIC code is freely available at Github and MATLAB File Exchange (link: [2]) and solves the general motion optimization problem by using the alternating direction method of multipliers (ADMM) [3]. Finite-element-based global DIC code is also available at MATLAB File Exchange [4].

We demonstrate that our ALDIC algorithm has high accuracy and precision while maintaining low computational cost, and is a significant improvement compared to current local and global DIC methods. For a review of both local and global DIC methods, and details of this new proposed ALDIC method, please see our paper [1] (full text can also be requested at [5]).

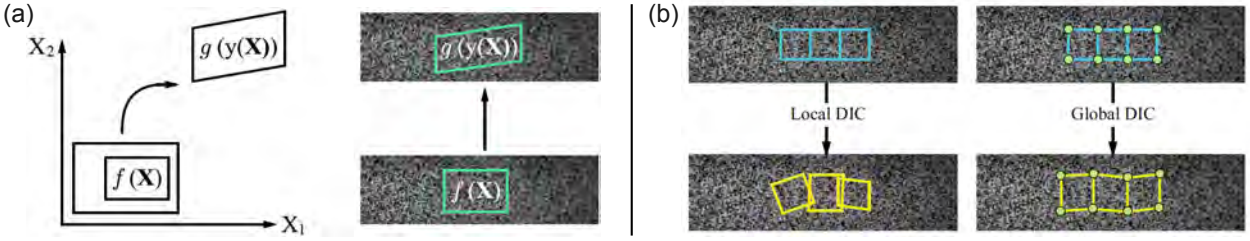


Figure 1: (a) Schematic showing a DIC reference image f(**X**), with a general speckle pattern, deforming into the deformed image g(**y**(**X**)) under some mapping **y**. **X** and **y** coordinates are in the reference and deformed images, respectively. (b) A schematic comparison between the local DIC method (left), where all the subsets are analyzed independently, and the finite-element-based global DIC method (right), where a global basis set is used to represent the full-field deformation.

Here some advantages of ALDIC algorithm are highlighted:

(i) It's a fast algorithm using distributed parallel computing for a global nonconvex optimization.

(ii) Global kinematic compatibility is added as a global constraint in the form of augmented Lagrangian, and solved using Alternating Direction Method of Multipliers (ADMM) scheme.

(iii) Both displacement fields and affine deformation gradients are correlated at the same time.

(iv) No need of much art-of-work about choosing displacement smoothing filters.

(v) It works well with compressed images and adaptive mesh [6, 7, 8].

(vi) It can solve an image sequence with multiple frames. Both *cumulative* and *incremental* DIC modes are implemented where the latter one is quite useful for measuring very large deformations.

(vii) If you are applying 3D volumetric digital volume correlation, please refer to our ALDVC paper [9, 10] and code [11, 12].

2 Code installation

ALDIC code can be downloaded at [2]. It has been tested on MATLAB versions later than R2018a. Parallel Computing Toolbox is highly recommended but not necessary to speed up the code.

To install the code, please download and unzip the code folder and put this folder on the MATLAB current working path.

After opening the main file "main_ALDIC.m", as shown in Fig. 2, ALDIC code can be executed by each section. Once you are familiar with the ALDIC code, you can execute the whole main file by clicking the "Run" button to run the whole file at once ("EDITOR >> RUN >>  "). ALDIC code is easy to modify based on user's custom parameter choice. In this code manual, we will introduce the ALDIC code section by section.

3 Code Section 1. MATLAB mex setup

Execute this section and we will try to build "mex" functions from C/C++ source codes for image grayscale value interpolation, where linear, bi-cubic (by default) and bi-cubic splines interpolations are implemented in this code. For example, by default we use bi-cubic interpolations where the associated mex set up file is called "ba_interp2.cpp" [13].

3.1 Test MATLAB mex setup

First, we test whether there is already a C/C++ compiler installed on your computer by inputting `mex -setup` and press Enter key on the MATLAB command window. If an available C/C++ compiler is already installed, please skip Section 3.2 and jump to Section 3.3.

3.2 Install mex C/C++ compiler

The step of installing mex C/C++ compilers is a common step for users to run C/C++ codes with MATLAB. Mac users usually don't come across the error message from mex C/C++ compilers. For Windows users, you can follow these steps to install mex C/C++ compiler. More details can be found in [14, 15].

- Download: TDM-gcc compiler from: http://tdm-gcc.tdragon.net/

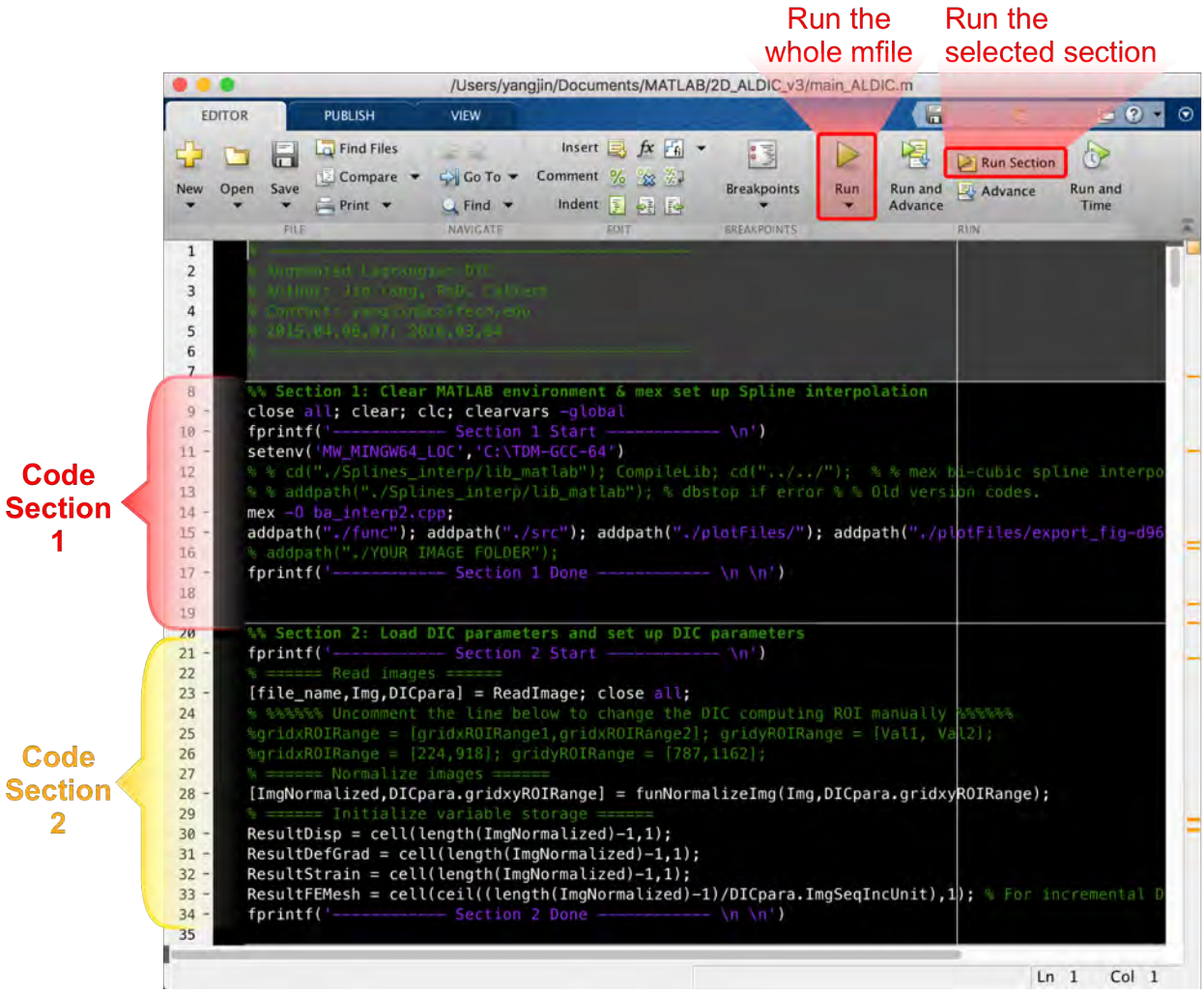


Figure 2: Main file of ALDIC code: "main_ALDIC.m". Each section can be executed in order by clicking the "Run Section".

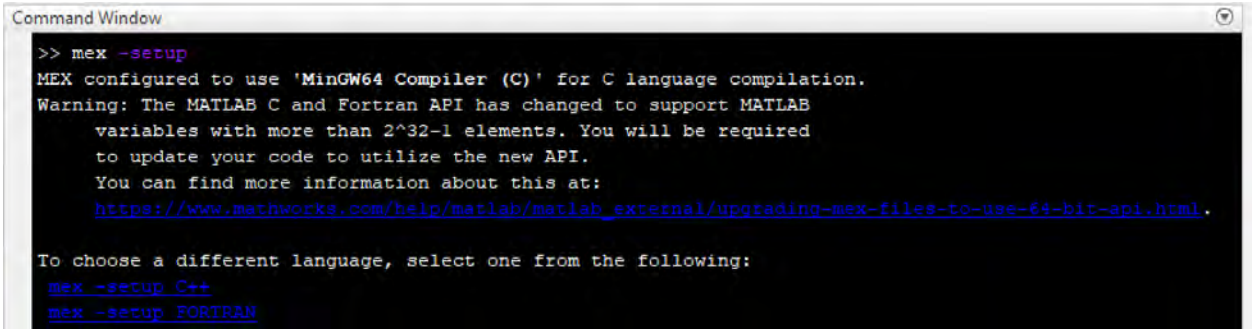


Figure 3: Message display on the command window when a mex C/C++ compiler is stalled successfully.

- Install TDM-gcc compiler on your computer. For example, I install it at 'C:\TDM-GCC-64' [1].
- Restart MATLAB and input these codes on the command window:
  `setenv('MW_MINGW64_LOC','YourTDMGCCPath'); mex -setup;`
  to check whether "mex" is set up successfully or not. Don't forget to replace the above `'YourTDMGCCPath'` using your own installation location of TDM-gcc package in the last step. For example, if it's installed at 'C:\TDM-GCC-64', please replace previous `'YourTDMGCCPath'` with 'C:\TDM-GCC-64'. If a mex C/C++ compiler is installed successfully, a message similar to (Fig. 3) will display on the MATLAB command window.

3.3 Execute code Section 1

Once a mex C/C++ compiler is installed, we can execute main_ALDIC.m code Section 1 and a successful message will display on the MATLAB command window, as shown in Fig. 3.

4 Code Section 2: Load images and set up DIC parameters & mesh

This section is to load both DIC reference and deformed images and set up DIC parameters. First, please put your images on the MATLAB working path. After executing this section, all the ALDIC associated parameters will be stored in the workspace structure variable "DICpara". All the mesh properties will be stored in the structure variable "DICmesh" after executing code Section 3. Here we make a short summary of both these two data structures in Table 1 and Table 2.

4.1 Load DIC images

MATLAB command screen displays these lines to allow users to select their choices to load DIC images, which will be explained in Sections 4.1.1-4.1.3.

[1] In practice, we find that this TDM-gcc compiler only works if installed on the first level main disks, such as 'C:\', 'D:\', 'E:\', etc.

Table 1: Summary of DIC parameters in “DICpara” structure

DICpara variable	DIC parameter	Description and comments
winsize	Subset window size [winsize_x, winsize_y]	Local subset size in ALDIC Subproblem 1
winstepsize	Subset window step [winstepsize_x, winstepsize_y,	The distance between neighboring local subsets, or the finite element size in ALDIC Subproblem 2.
gridRange	DIC region of interest (ROI)	ROI can be defined by clicking top-left and bottom-right corner points.
Subpb2-FDOrFEM	0-’FD’: Finite difference method; 1-’FEM’: finite element method	Both finite different and finite element methods are implemented to solve ALDIC Subproblem 2.
ClusterNo	Number of threads to perform parallel computing	ALDIC Subproblem 1 can be sped up by applying parallel computing.
ImgSize	Image size	Size of 2D DIC images
ImgSeqIncUnit	To decide to perform cumulative or incremental DIC mode	Cumulative DIC is always to compare with the first reference frame; incremental DIC could update the reference frame.
ImgSeqInc-ROIUpdate-OrNot	In incremental mode, ROI can be updated at the same time of updating reference image.	It’s recommended to manually update ROI at the same time of updating reference image for measuring large deformations.
NewFFTSearch	Method to update initial guess to solve an image sequence	Result of last frame can be assigned as the initial guess for the next frame.
SizeOfFFT-SearchRegion	Size of FFT-cross correlation search region in the initial guess section	The search region size should be larger than the maximum x- and y-displacements.
displayIterOrNot	Display convergence details of Subproblem 1 IC-GN iterations	0 or 1: Don’t print or print IC-GN convergence info of each local subset.
Subpb1ICGN-MaxIterNum	Maximum IC-GN iterations to solve Subproblem 1	Subsets fail to converge within ’Subpb1ICGNMaxIterNum’ iteration steps will be marked as bad subsets.
ICGNtol	Tolerance threshold of IC-GN iterations in solving Subproblem 1	E.g., ICGNtol takes value of 10^{-4} px by default.

Table 2: Summary of DIC mesh in “DICmesh” structure

DICmesh variable	DIC parameter	Description and comments
coordinatesFEM elementsFEM	Coordinates of nodal points in the finite element mesh and their connectivity	Linear 4-node quadrilateral (Q4) elements are used here. However, it can be extended to other type of finite elements with arbitrary shape function.
dirichlet neumann	FE-mesh nodal points at the boundary	Indices of nodal points at ROI borders are assigned with Dirichlet or Neumann boundary conditions.
x0, y0	Regular FE-mesh nodal grids	Here we use uniform regular Q4 FE-mesh.

```
------------ Section 2 Start ------------
Choose method to load images:
0: Select images folder;
1: Use prefix of image names;
2: Manually select images.
Input here:
```

One comment about the ALDIC code is that it always manipulate the deformed images and attempts to transform them back to the reference image to compute their deformation fields which is based on the *Lagrangian* description (see more details in Appendix 15.3.5). If the user wants to track the deformation field in the *Eulerian* description, he/she can select the reference image as the second image, choose the deformed image to be the first, and manipulate the reference image to transform to the current deformed image.

4.1.1 Load DIC images from a selected folder

If we choose the method `0: Select images folder` to load the DIC images, the user will be asked to select the folder path, and all the images in that folder will automatically be loaded. In *cumulative* DIC mode, the first frame is set to be the fixed reference image by default, while all the subsequent frames in the image sequence are set to be the deformed images whose deformations will be tracked in the Lagrangian description. In *incremental* DIC mode, the reference image can be updated after every specific number ("DICpara.ImgSeqIncUnit") of frames. After loading DIC images, please jump to Section 4.2.

4.1.2 Load DIC images with prefix of image name

If we choose the method `1: Use prefix of image names` to load DIC images, the user also needs to provide prefix text words of their DIC images (and all of these images need to be added to the MATLAB path as well). For example, if all DIC images are named in the following format: "img_0001.tif", "img_0002.tif", $\cdots$, the user should input " `img_0*.tif` " on the MATLAB command screen to load all the DIC images started with prefix "img_0" and in the "tif" format. After loading DIC images, please jump to Section 4.2.

```
What is prefix of DIC images? E.g. img_0*.tif.
Input here:
```

4.1.3 Load DIC images manually

If we choose the method `2: Manually select images` , the user needs to load DIC images frame by frame manually, as shown in Fig. 5. By default, the ALDIC code sets the first image as the reference, undeformed image and all other images as deformed images.

```
--- Please load first image ---
--- Please load next image ---
Do you want to load more deformed images? (0-yes; 1-no)
```

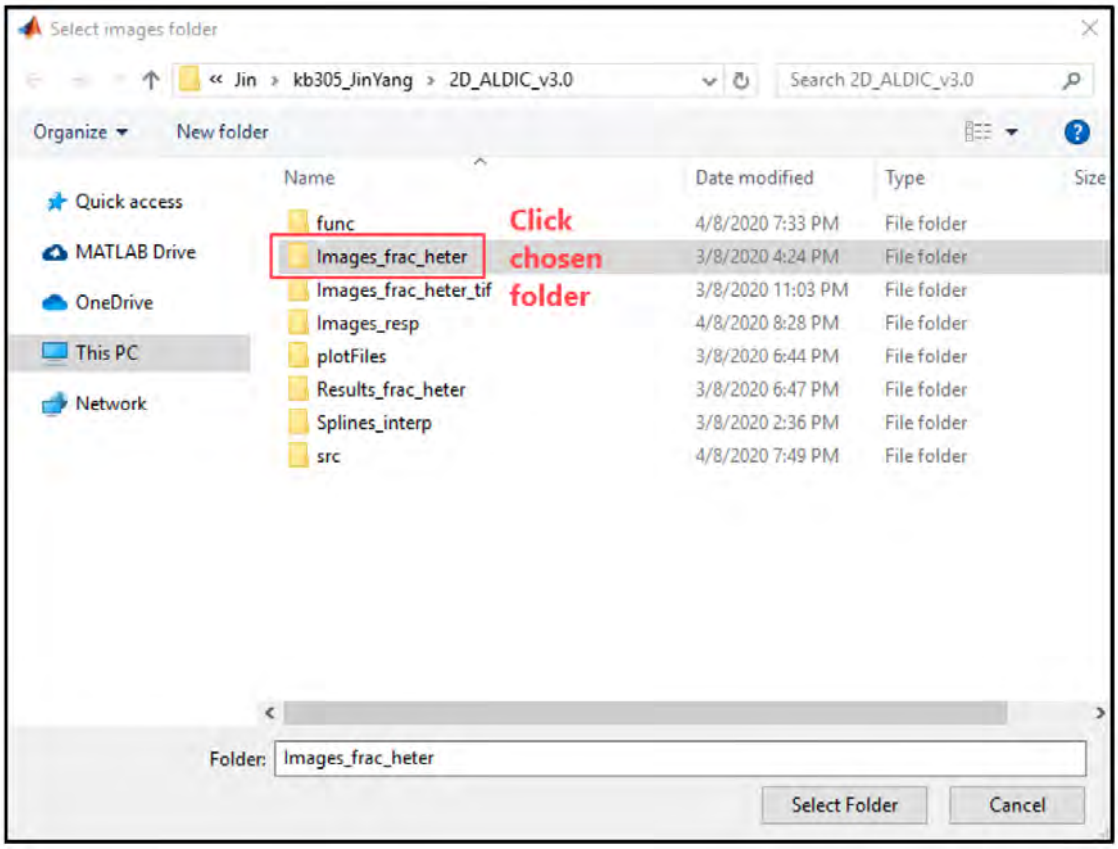


Figure 4: Load DIC images by selecting the folder including DIC images. For example, after clicking chosen folder "Images_frac_heter", then all the images in this folder will be loaded automatically.

Input " `0` " if you want to continue uploading images and input " `1` " if you want to stop uploading images. For example, the user can select the first reference image as "img_0000.tif", and select a second image as "img_0570.tif".

4.2 Define region of interest (ROI)

Execute code Section 2, the user can click both the top-left and bottom-right corner points on a popped-out image to define DIC region of interest (ROI). On the command window, it will display:

```
--- Define ROI corner points at the top-left and the bottom-right ---
```

Then the user could first click a left-top point and click a right-bottom point to define ROI directly on the DIC image, see Fig. 6.

After clicking both the top-left and bottom-right corner points, the command window screen will display their coordinates in the unit of pixels. E.g., in the image shown in Fig. 6, I define a ROI with top-left and bottom-right corner points are:

```
Coordinates of top-left corner point are (322.786,74.730)
```

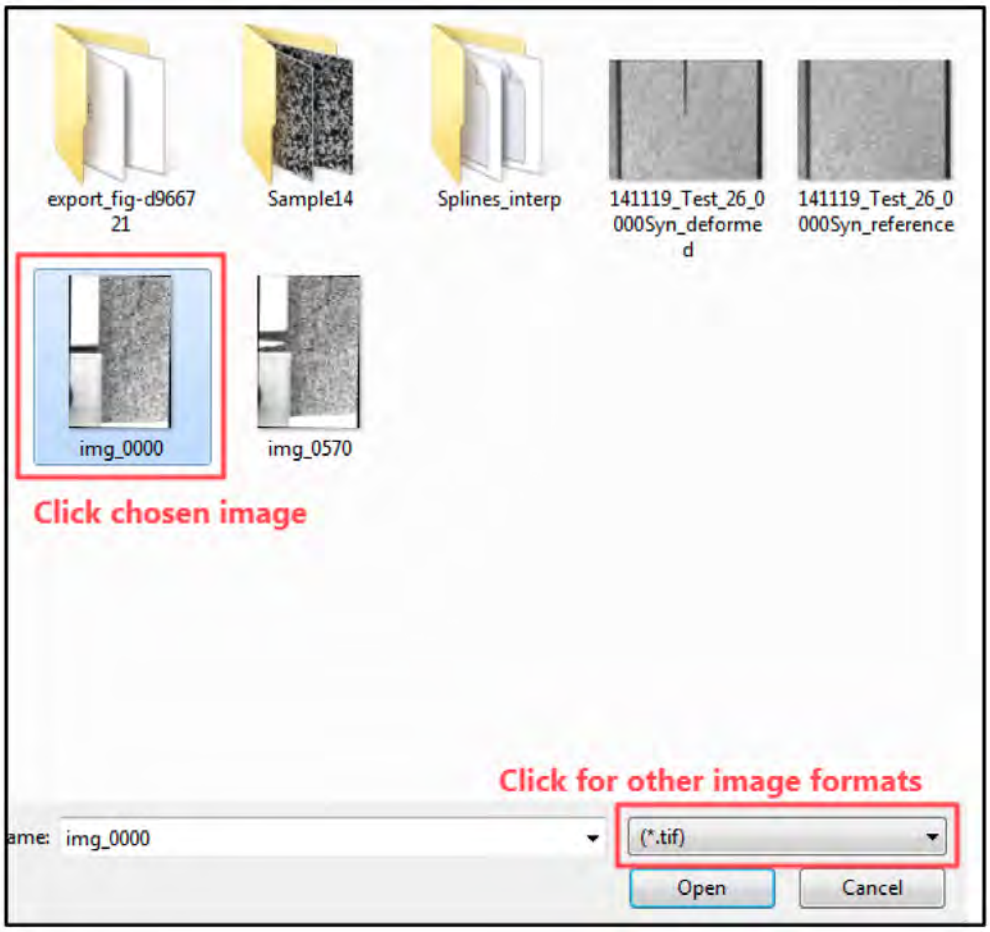


Figure 5: Manually load DIC images by clicking both reference and deformed images.

```
Coordinates of bottom-right corner point are (750.063,1128.439)
```

Comment 1: If the top-left or bottom-right corner point is clicked out of the image, it will be adjusted to the nearest point on the original DIC image border automatically.

Comment 2: If user prefers to use command lines to input ROI coordinates instead of directly clicking corner points on the DIC image. Please uncomment these lines

```
\% gridxROIRange = [gridxROIRange1, gridxROIRange2];
\% gridyROIRange = [gridyROIRange1, gridyROIRange2];
```

,

and modify values of ”gridxROIRange” and ”gridyROIRange”.

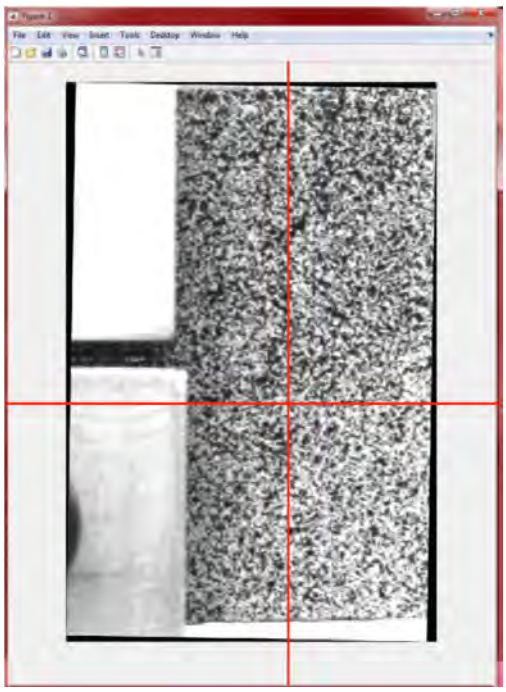


Figure 6: Manually click top-left and bottom-right corner points to define DIC region of interest (ROI).

4.3 Set up DIC parameters

Then the user will be asked to determine the size of the local subset and the subset spacing. "Subset size" is the edge length of the subset window (needs to be an even integer, 2M, and the subset set window is located from -M px to M px corresponding to the subset center); while the "subset step" is the distance between two adjacent subsets and can be chosen independently from the subset size.

```
--- What is the subset size? ---
Input here: 20
--- What is the subset step? ---
Input here: 10
```

Subset step size is also the finite element mesh size in the ALDIC Subproblem 2 global step. Practically the user can choose subset size from 10 px by 10 px to 60 px by 60 px to include 3∼5 features of your DIC images; The subset step size can be chosen as 0.25∼1 times of the subset size. E.g., in the heterogeneous fracture example "img_0000.tif" and "img_0570.tif", we choose subset size as `20` px, and subset step as `10` px.

We also need to choose the solver method for ALDIC Subproblem 2 (global step). For this version of the code, we can only work with a uniform grid mesh (see Sections 13-14 to solve complex geometry where quadtree meshes are applied), and there is little difference whether you choose the "Finite difference method" or the "Finite element method". Even if you choose the "Finite difference method", there are still finite element mesh generated (cf "DICmesh" in the Matlab

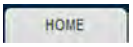
workspace) which could help you conduct FEA analysis if you want to combine DIC with other FEA codes/software afterwards. My personal experience is that finite difference method is faster and more accurate than the finite element method in the ALDIC Subproblem 2 (global step) due to the boundary effects.

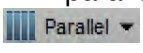
```
--- Method to solve ALDIC global step Subproblem 2 ---
    1: Finite difference(Recommended)
    2: Finite element method
Input here: 1
```

We need to set up parallel pools or tell MATLAB we don't want to use parallel pools.

```
--- Set up Parallel pool ---
How many parallel pools to open? (Put in 1 if no parallel computing)
Input here: 4
```

If we don't want to use parallel pools, please input `0` or `1` . If we want to use parallel computing, put the number of your parallel pools. E.g., Input `4` .

MATLAB parallel computing environment can be set in "Home 

≫ Environment ≫ Parallel 

≫ Parallel Preferences" (see Fig. 7 and [16] for more information about `parpool`).

4.4 Additional parameters setup when dealing with image sequence

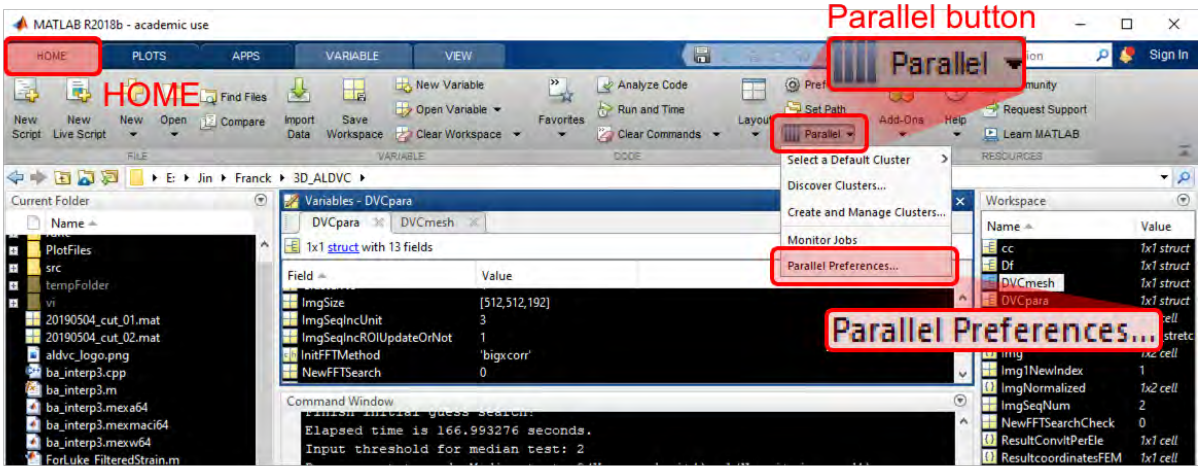
If we upload an image sequence of more than two frames, for each new frame we can choose to use the displacement results from the last frame as an initial guess of the new image frame, or we can redo FFT-based cross correlation optimization to compute a new initial guess for each new frame.

This selection also depends on how large the relative displacement between two consecutive frames. In general, if the relative displacement field between two consecutive frames is less than $5 \sim 7$ pixels, the deformation result of last frame can be used as the initial guess for the new frame. Otherwise, it is suggested that we still need to redo FFT-based cross correlation optimization.

```
Since we are dealing with an image sequence with multiple frames, for each
    new frame, do we use the result of last frame as an initial guess or
   redo FFT initial guess for every new frame?
0: Use last frame;
1: Redo initial guess.
Input here:
```

When post-process an image sequence with more than two frames, the user could decide to perform either *cumulative* mode DIC or *incremental* mode DIC. Cumulative mode is the default setting, and all the following frames will be compared with the first frame. However, incremental mode is preferred when dealing with extremely large deformations, but some accuracy may be lost as the reference image has been updated. We recommend that the user tries cumulative mode first, and then try incremental mode if cumulative mode doesn't work well.

(a)



(b)

Figure 7: Set up MATLAB parallel preferences. See [16] for more information about `parpool` .

```
--- Choose cumulative or incremental mode ---
     0: Cumulative(By default);
     1: Incremental;
Input here: 1
```

If the user chooses to use incremental mode, he/she will further be inquired to input how often to update the reference image:

```
Incremental mode: How many frames to update reference image once?
Input here: 10
```

E.g., I want to update my reference image every 10 frames, so I input: `10` . The smallest number you can enter is `1` , which means that reference image updates every frame.

Every time the reference image is updated, you can choose to update the region of interest (ROI) at the same time. To achieve this, user will be asked as follows:

```
Update ROI at the same time of updating reference image?
    0: Do not update ROI;
    1: Manually(Recommended);
    2: Automatically;
Input here: 1
```

E.g., Input `1` or `2` if you want to update ROI at the same time of updating reference image. In theory, this ROI update can be done automatically by deforming the last frame. However, I found that manual updating of ROIs is still the most robust.

5 Code Section 3: Computing an initial guess of unknown deformation from FFT-based cross correlation

5.1 FFT-based methods to compute initial guess

In this section, the initial estimate of the unknown displacement is calculated by maximizing the (zero-normalized) cross correlation function C_{ZNCC}, see Fig. 8(a-b).

$$C_{ZNCC} = \frac{\int (f-\mu_f)(g-\mu_g)}{\sigma_f \sigma_g} \tag{1}$$

where μ_f, μ_g are average grayscale values of DIC images, f and g; σ_f, σ_g are the standard deviation of image grayscale values. Above cross correlation can be computed fast using Fourier transform convolution theorem and fast Fourier transform (FFT). For example:

$$C_{CC} = \int f(\mathbf{x}) g(\mathbf{x}+\mathbf{u}) d\mathbf{x} = \mathcal{F}^{-1}\left[\overline{\mathcal{F}[f(\mathbf{x})]} \circ \mathcal{F}[g(\mathbf{x}+\mathbf{u})]\right] \tag{2}$$

where $\mathcal{F}[\cdot]$ denotes the Fourier transform, $\overline{(\cdot)}$ denotes the complex conjugate, and "$\circ$" denotes the Hadamard product (entry-wise product) and the absolute values are taken entry-wise as well.

The ALDIC code provides three ways to compute the initial guess for an unknown displacement field, as shown in Fig. 8(c-e), and following lines will appear on the screen:

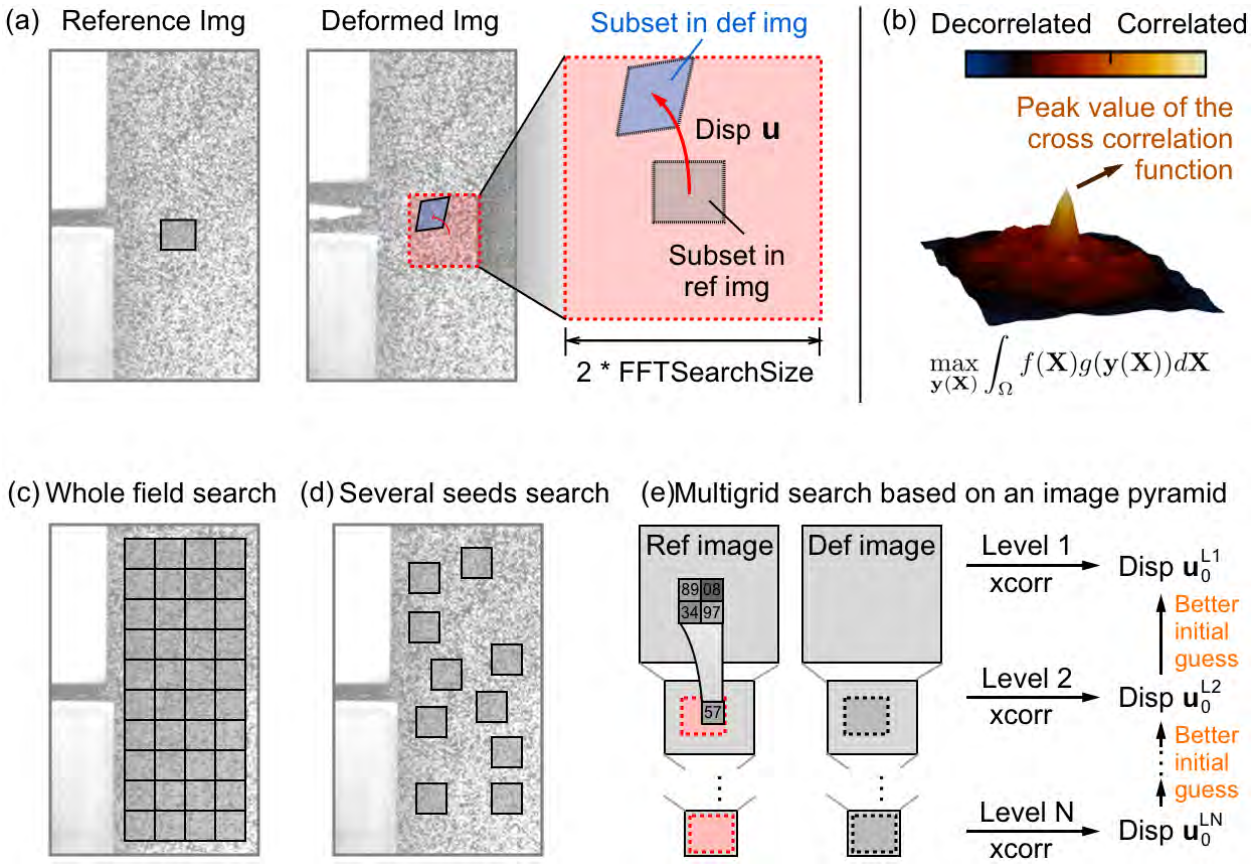


Figure 8: (a-b) The initial estimate of the unknown displacement of a local subset is calculated by maximizing the (zero-normalized) cross correlation function. (c-d) All local subsets or a few manually clicked seeds are analyzed using FFT-based cross correlation to compute the initial guesses of their deformations. (e) A multigrid method based on the constructed image pyramids is proposed to compute the initial estimate of full-field deformation.

```
--- Method to compute an initial guess of displacements ---
    0: Multigrid search based on an image pyramid
    1: Whole field search for all the subsets
    2: Search near manually clicked seeds and then interpolate for the
   full-field
Input here:
```

Method 1 (whole field search, Fig. 8(c)) executes FFT-based cross correlation for all local subsets (subsets can overlap), while **Method 2** (Several seeds search, Fig. 8(d)) is to compute near several manually clicked local seeds. Both **Methods 1 & 2** will ask the user to enter the size of the search area. The user should try to search for a small value of the search area size and incrementally increase that value until it exceeds the maximum x- and y-displacements. For example, in the heterogeneous fracture example as shown in Fig. 10, the max magnitude of the displacements are around 60 px, thus the search area size can be chosen as an integer larger than `60` . If the chosen FFT search area size is too small, for example, `30` or `50` , the computed initial displacements

will be noisy (see Fig. 9(a-b)). If this happens, the user needs to continually increase the size of the search area until a good initial guess is found. For example, search area size equals `60` in Fig. 9(c).

```
--- What is your initial guess search zone size (pixels)? ---
User should start to try a small integer value, and gradually increase the
value of the search zone size until it is larger than the magnitudes of
|disp u| and |disp v|.
Input here: 60
Finish initial guess search!
```

To further speed up the above FFT-based optimization process, we also provide a multigrid **Method 0** based on the constructed image pyramids (see Fig. 8(e)). There is no need to provide the search area size in this Method. For example, an initial guess of the above heterogeneous fracture deformation is solved and shown in Fig. 10.

5.2 Remove noise in computed initial guess

In practice, FFT-based cross correlation method solved initial guess [2] may have large noise, the user can further remove these bad points by applying a median filter, setting a q-factor threshold, and setting upper and lower limits of displacements. Users can also continue to delete the bad points directly by clicking them in each image and then press the " `Enter` " key. However, this step requires some manual work. If noise elimination is necessary, the user can follow the following steps:

```
Do you clear bad points by setting upper/lower bounds once more?
(0-yes; 1-no)
```

If the user inputs " `0` ", then he/she will be asked to enter the upper and lower limits of the x- and y-displacements. If the user inputs " `1` ", this process will be skipped.

```
% ======= Find bad initial guess points manually by setting bounds =======
What is your upper bound for x-displacement?
What is your lower bound for x-displacement?
What is your upper bound for y-displacement?
What is your lower bound for y-displacement?
Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
```

Besides setting upper and lower bounds to remove local bad points, we can continue removing bad points directly by clicking them in the image.

```
% ======= Find bad initial guess points manually =======
'Do you clear bad points by directly pointing x-disp bad points?
(0-yes; 1-no)';
'Do you clear bad points by directly pointing y-disp bad points?
(0-yes; 1-no)';
```

[2] Some designed filters or iterative deformation method (IDM) can further improve the accuracy of initial guess, cf [17, 18, 19, 20]. These are beyond the scope of this code manual. In addition, accuracy of computed initial guess will further be improved after executing ALDIC ADMM iterations (code Sections 4-6).

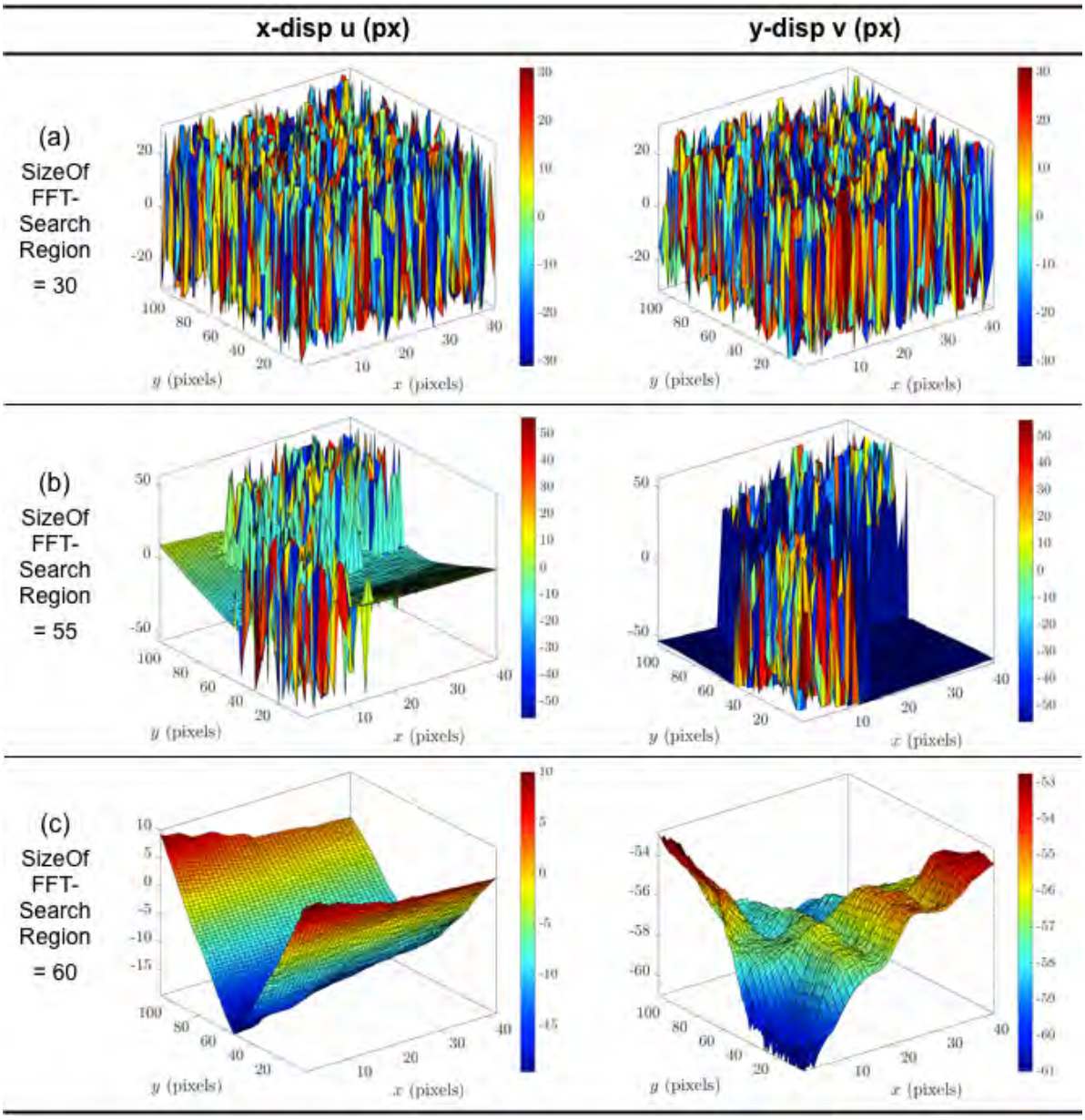


Figure 9: **Method 1** solved displacement fields by the FFT-based cross correlation method with different search area sizes. (a-b) If the chosen size of the search area is too small, computed initial displacements will be super noisy. (c) A good "search area size" should not be smaller than the magnitudes of displacement components (|dispu|, |dispv|).

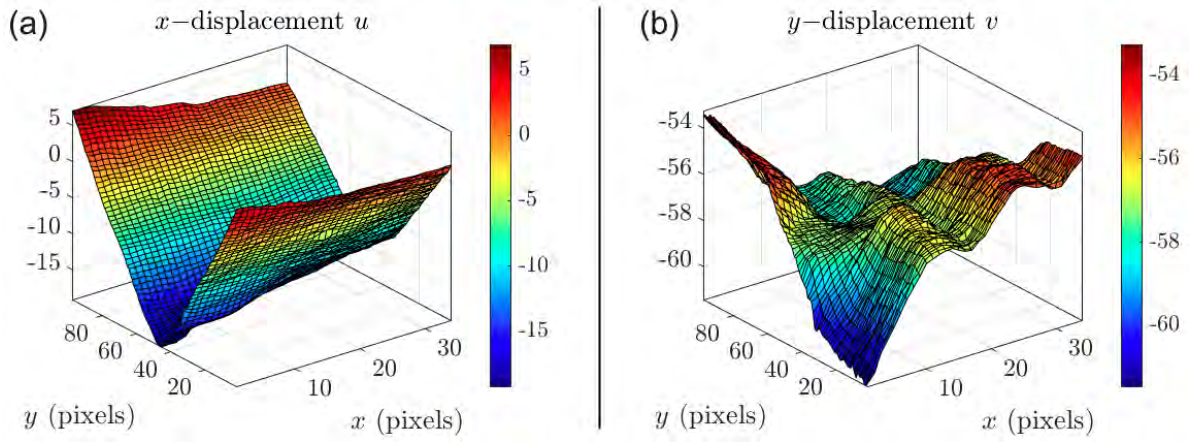


Figure 10: **Method 0** – Multigrid FFT-based cross correlation solver – computed initial estimate of the unknown deformation field.

Click all the bad points directly, and then press the " `Enter` " key.

At the end of this section, a finite element mesh is also automatically generated.

```
--- Finish setting up mesh and assigning initial value! ---
```

The image pixel grayscale gradient can also be calculated fast by using a finite difference operator and convolution operation.

```
--- Start to compute image gradients ---
--- Computing image gradients done ---
```

6 Code Section 4: ALDIC Subproblem 1 (first local step)

6.1 Local subset IC-GN solver

In this section, we use IC-GN (inverse compositional-Gauss Newton) scheme to solve ALDIC ADMM Subproblem 1 (local step), where distributed parallel computing has been used. After Executing this section, the IC-GN solver will work and a waiting bar will pop out and allow the user to visualize when a program will compute the ALDIC Subproblem 1, as shown in Fig. 11. If the DIC image size is very large, MATLAB parpool can take a few minutes before initializing parallel computing and transferring all image datasets. So, if you don't see a wait-bar coming right away, do not worry and wait a little longer.

```
------------ Section 4 Start ------------
***** Start step1 Subproblem1 *****
Local ICGN bad subsets %:
Elapsed time is xxx seconds.
```

(a)

```
------------ Section 4 Start ------------
***** Start step1 Subproblem1 *****
Starting parallel pool (parpool) using the 'local' profile ...
connected to 4 workers.
```

(b)

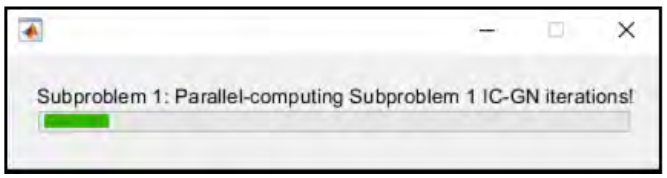


Figure 11: (a) Code section 4 starts and 4 threads are connected. (b) A wait-bar pops out and allows the user to visualize when a program will finish solving the ALDIC Subproblem 1.

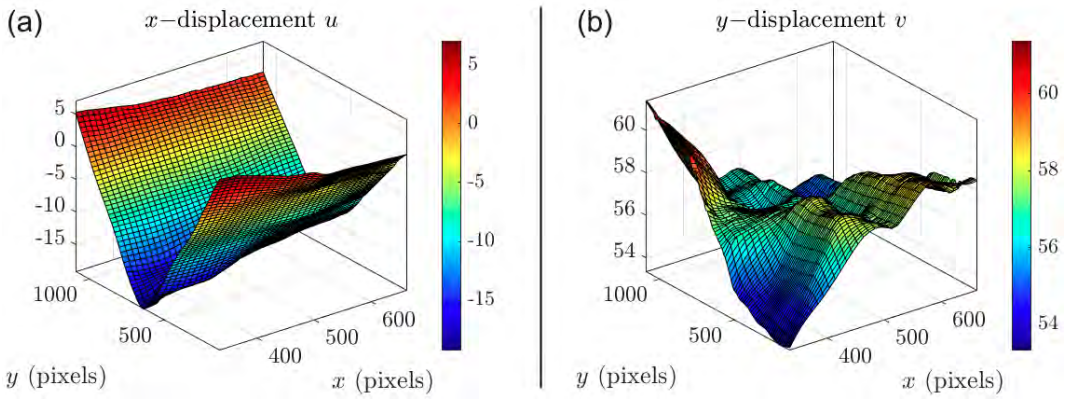


Figure 12: Solved displacements from the ALDIC subproblem 1 first local step.

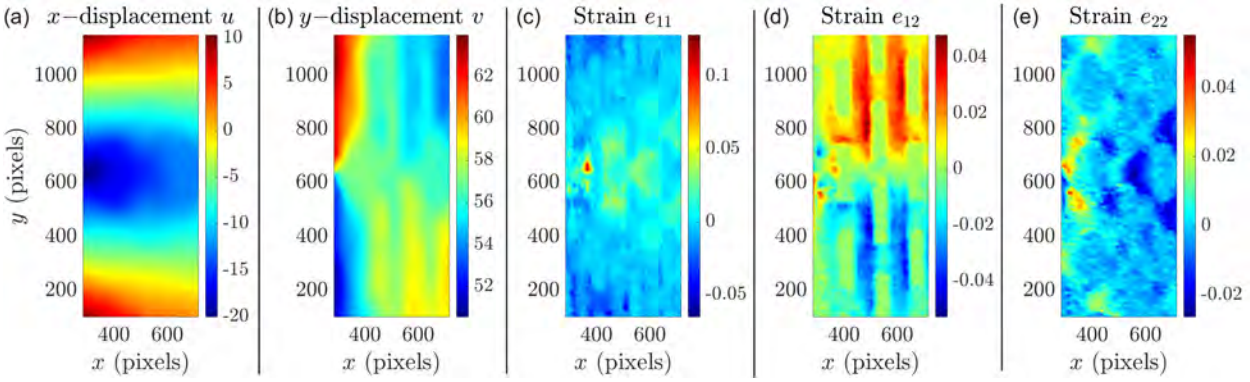


Figure 13: Contour plots of solved displacements and infinitesimal strains from the ALDIC subproblem 1 first local step.

6.2 Remove results of IC-GN bad subsets

After this step completes, a report will display on the command window. Similar to Section 5, users can further eliminate all bad points by applying a median filter, setting a q-factor threshold, and setting both upper and lower bounds of the displacement. Users can also continue to directly delete bad points by clicking them in each image and then press the " `Enter` " key.

```
--- Start to remove bad local points ---
% ======= Find bad Local IC-GN step points manually by pointing them
   =======
'Do you clear bad points by directly clicking x-disp bad points? (0-yes;
   1-no)';
'Do you clear bad points by directly clicking y-disp bad points? (0-yes;
   1-no)';

% ======= Find bad guess points manually by setting bounds =======
What is your upper bound for x-displacement? % User input
What is your lower bound for x-displacement? % User input
What is your upper bound for y-displacement? % User input
What is your lower bound for y-displacement? % User input
Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no) %
   Input "0" if you want to redo it.

--- Remove bad points done ---
------------ Section 4 Done ------------
```

Comment: If you see this error message:

```
'Insufficient data for surface estimation.','[u1temp] = gridfit(
   coordinatesFEM (notnanindex,1), coordinatesFEM(notnanindex,2),U(2*
   notnanindex-1), Coordxnodes, Coordynodes,'regularizer','springs');
   u1temp = u1temp';'and 'main_ALDIC (line 106) LocalICGN(U0,DICmesh.
   coordinatesFEM,Df,fNormalized,gNormalized,DICpara,'GaussNewton',tol);'.
```

Or if you see that all the local subsets diverged, in another word, the percentage of local ICGN bad subset is 100%, this usually happens because of the bad DIC parameter choice or bad DIC pattern quality. Here are a few steps that may help improve poor convergence, see FAQ [15.3.3].

7 Code Section 5: ALDIC Subproblem 2 (first global step)

7.1 Comparison between finite difference and finite element methods in solving Subproblem 2

After solving the ALDIC ADMM Subproblem 1 (local step), we solve the ADMM Subproblem 2 (global step) in this section. Both finite difference and finite element solvers are implemented. Both finite difference and finite element methods work well for uniform regular grid meshes. Finite element method is easiler to implement for arbitrary geometry and can be combined with other adaptive mesh techniques, see Sections 13-14.

For uniform grid meshes (see Sections 13-14 to solve complex geometry where quadtree meshes are applied), there is little difference whether you choose the "Finite difference method" or the "Finite element method". Even if you choose the "Finite difference method", there are still finite element mesh generated (cf "DICmesh" in the Matlab workspace) which could help you conduct FEA analysis if you want to combine DIC with other FEA codes/software afterwards. My personal experience is that finite difference method is faster and more accurate than the finite element method in the ALDIC Subproblem 2 (global step) due to the boundary effects.

7.2 Finite difference method

To solve the ADMM Subproblem 2 by the finite difference method, a sparse finite difference operator **D** is built by `funOperator3` satisfying:

$$\{\mathbf{F}\} = \mathbf{D}\{\mathbf{u}\} \tag{3}$$

where **u** is displacement, and **F** is "deformation gradient tensor minus identity", whose associated components related to node (i) are

$$\begin{aligned} \mathbf{F} &= \left\{ \cdots \quad F_{11}^{(i)} \quad F_{21}^{(i)} \quad F_{12}^{(i)} \quad F_{22}^{(i)} \cdots \right\}^T \\ \mathbf{u} &= \left\{ \cdots \quad u_1^{(i)} \quad u_2^{(i)} \quad \cdots \right\}^T \end{aligned} \tag{4}$$

```
------------ Section 5 Start ------------
***** Start step1 Subproblem2 *****
Assemble finite difference operator D
Elapsed time is xxxxxx seconds.
Finish assembling finite difference operator D
***** Start step1 Subproblem2 *****
Elapsed time is xxxxxx seconds.
------------ Section 5 Done ------------
```

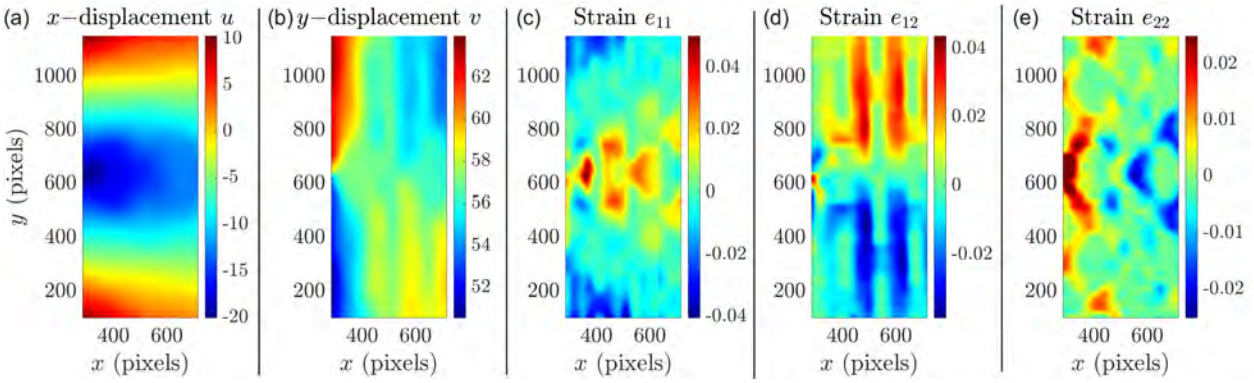


Figure 14: Contour plots of solved displacements and infinitesimal strains from ALDIC subproblem 2 first global step.

7.3 Finite element method

Subproblem 2 can also be solved by the finite element method. In this code, we use a 4-node quadrilateral (Q4) finite element mesh, the same element type with our global DIC code (finite element based global DIC code is also available at MATLAB File Exchange for comparison [4]).

8 Code Section 6: ALDIC ADMM iterations

Then, in this section, ALDIC will perform alternating direction method of multipliers (ADMM) iterations. The tolerance threshold of ADMM iteration is set to be `1e-4` by default. But this threshold can be manually adjusted by modifying `tol2` . In practice, ALDIC ADMM usually converges within 3$\sim$5 iteration steps for the update less than `1e-2` px. For example, in the case of heterogeneous fracture example, the ADMM iteration converges at step 6, but there is little difference between the solution in step 3, as shown Fig. 15.

```
------------ Section 6 Start -------
***** Start step2 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 6.078113 seconds.
***** Start step2 Subproblem2 *****
Elapsed time is 0.007632 seconds.
Update local step = 0.018188
Update global step = 0.028851
**********************************

***** Start step3 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 6.036331 seconds.
***** Start step3 Subproblem2 *****
Elapsed time is 0.007165 seconds.
Update local step = 0.0027562
Update global step = 0.025722
**********************************

***** Start step4 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 5.937378 seconds.
***** Start step4 Subproblem2 *****
Elapsed time is 0.007198 seconds.
Update local step = 0.0019051
Update global step = 0.0018707
**********************************

***** Start step5 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 5.892075 seconds.
***** Start step5 Subproblem2 *****
```

```
Elapsed time is 0.007503 seconds.
Update local step = 0.00015808
Update global step = 0.00027234
**********************************

***** Start step6 Subproblem1 *****
Local step bad subsets total # is: 0
Elapsed time is 6.106291 seconds.
***** Start step6 Subproblem2 *****
Elapsed time is 0.007841 seconds.
Update local step = 2.2404e-05
Update global step = 4.8639e-05
**********************************
```

9 Code Section 7: Check convergence

This section checks the convergence of the ALDIC ADMM iterations and removes some temporary variables from the workspace. If you don't need this section, skip it.

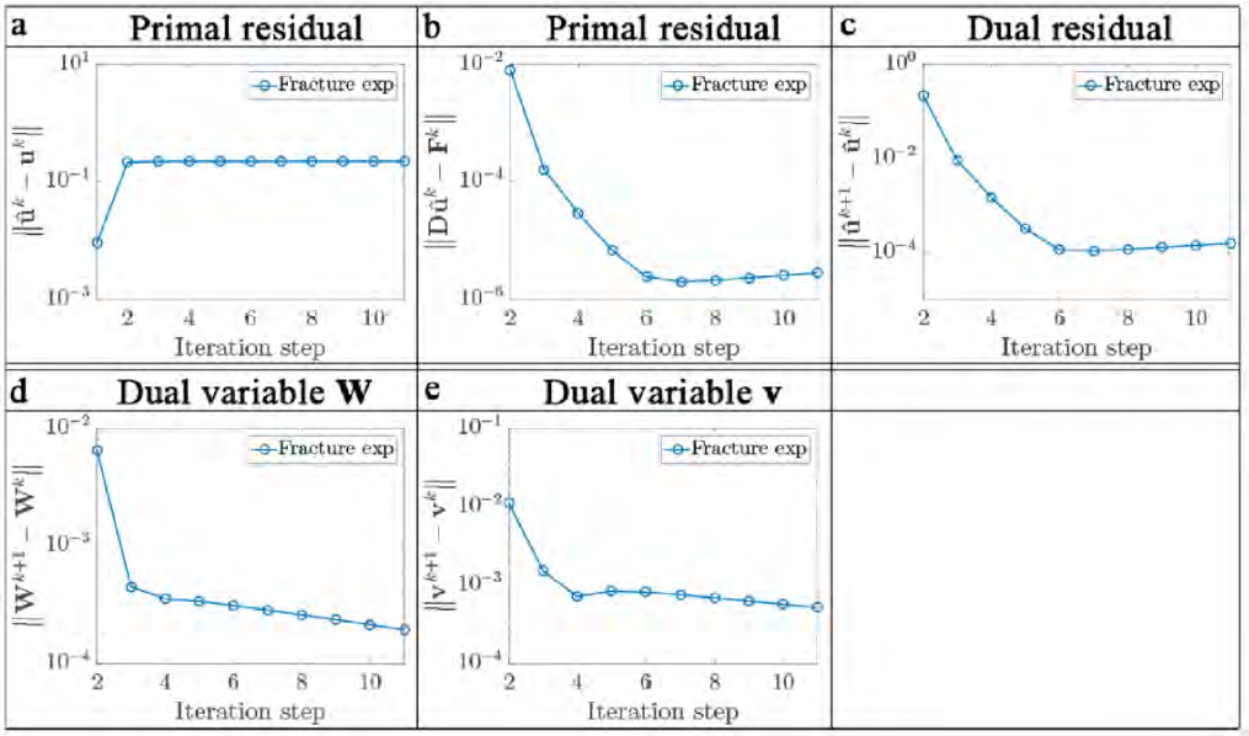


Figure 15: ALDIC ADMM algorithm usually converges within $3 \sim 5$ iterations.

10 Code Section 8: Compute strains

10.1 Smooth displacement field if needed

Before computing strains, solved displacement fields can be further denoised if necessary. In most cases, the ALDIC solved displacement fields are already denoised, so further smoothing solved displacement fields is usually no longer required.

```
------------ Section 8 Start ------------
Do you want to smooth displacement? (0-yes; 1-no)1
```

If you enter " `0` ", a Gaussian smoothing filter with a standard deviation of 0.5 will be applied. If a stronger smoothing filter is needed, please edit the Gaussian filter parameters " `DICpara.DispFilterSize` ", and " `DICpara.DispFilterStd` ".

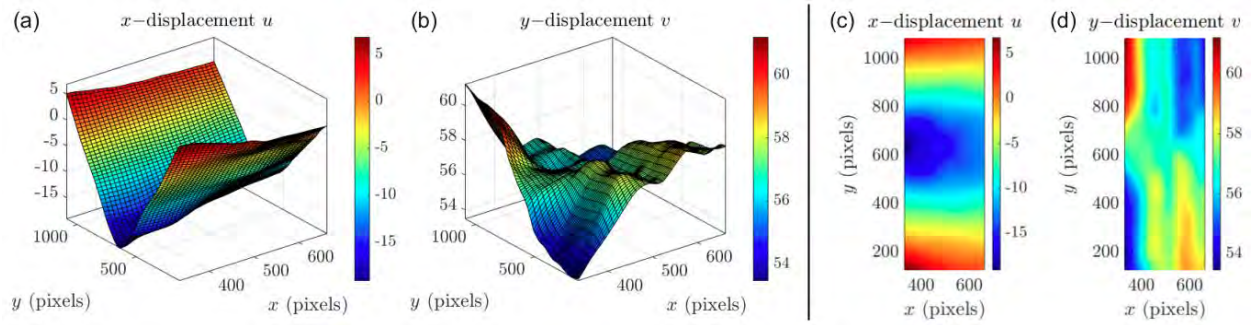


Figure 16: The ALDIC solved displacement fields are much already denoised and usually there is no need to apply additional smoothing filters anymore.

10.2 Compute strain field

In the ALDIC, "deformation gradient minus identity" **F** is a direct output besides displacement field. The strain field can also be calculated from numerically differentiating solved displacement field. We implement four methods to calculate the strain field.

(0) First method is a **direct output** from ALDIC solved **F** (deformation gradient minus identity);

(1) **Central finite difference** of the solved displacement field;

(2) **Plane fitting method** to differentiate the solved displacement field. In this method, you will be asked to input half plane width to define the size of the fitted plane;

(3) Strain field can also be computed from **finite element Gauss points**.

```
What method to use to compute strain?
    0: Direct output from ALDIC;
    1: Finite difference(Recommended);
    2: Plane fitting;
```

```
    3: Finite element;
Input here: 1
```

If the user enter " `2` ", MATLAB command window will display following lines to continue asking about the size of the fitting-plane:

```
What is your half window size: % Input half window size for plane fitting
```

Three popular types of strains are implemented in the ALDIC code: infinitesimal strain, Eulerian strain based on deformed configuration, and finite Green-Lagrangian strain.

```
Infinitesimal stran or finite strain?
    0: Infinitesimal stran;
    1: Eulerian strain;
    2: Green-Lagrangian strain;
    3: Others: code by yourself;
Input here: 0
```

E.g., the comparison between different methods for calculating the strain fields of heterogeneous fracture data set is shown in Fig. 17, where results of this example are stored in " `./results_frac_heter/` ".

10.3 Plot and save results

Then the user will be asked to choose to plot the solved deformation fields on the undeformed, reference configuration (see Fig. 18), or to plot over the current, deformed configuration (see Fig. 19).

```
Over which image(s) you want to plot the results?
    0: First image only (if you are using only two images);
    1: Second and next images;
Input here: 1

Save figures into the format:
1: jpg(Choose transparency 0~1)
2: pdf(Choose transparency = 0)
3: Others: Edit codes in ./plotFiles/SaveFigFiles.m
Input here:
```

In the current version of the ALDIC code, to overlay with the original DIC images, final plots can only be saved in the "jpg" format.

```
Define transparency for overlaying original images:
Input a real number between 0(Only original images)
and 1(Non-transparent deformation results).
Input here(e.g. 0.5):
```

Finally, don't forget to save results for future use. All the DIC results will be saved in a Matlab matfile by executing following codes, see Table 3.

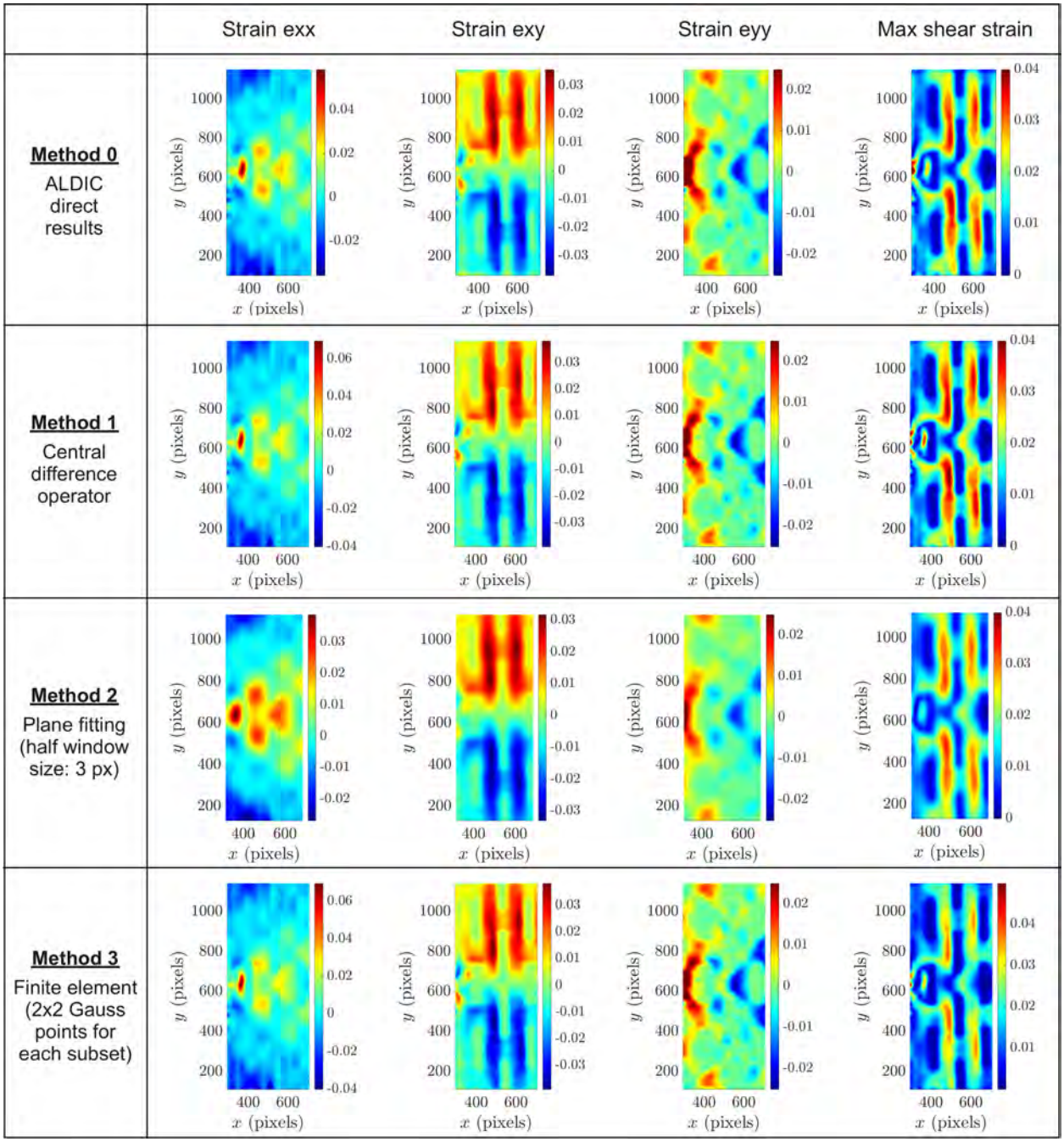


Figure 17: Comparison between different methods to compute strain fields for heterogeneous fracture example.

```
results_name = ['results_',imgname,'_ws',num2str(DICpara.winsize),'_st',
    num2str(DICpara.winstepsize),'.mat'];
save(results_name, 'file_name','DICpara','DICmesh','ResultDisp','
    ResultDefGrad','ResultStrain','ResultFEMesh',...
     'ALSub1Time','ALSub2Time','ALSolveStep');
```

Table 3: Summary of DIC results

Variable	Description
DICpara	ALDIC parameters, see Table 1
DICmesh	ALDIC finite element mesh details, see Table 2
ResultDisp	Solved displacements from ALDIC
ResultDefGrad	Solved "deformation gradient minus identity" **F** from ALDIC
ResultStrain	Solved strains from ALDIC code Section 8
ResultFEMesh	Stored FE-mesh due to reference frame update in incremental mode
ALSub1Time, ALSub2Time	Computation time in solving Subproblems 1 & 2

11 Code Section 9: Compute stress

If the material constitutive relation of the specimen is known, the relevant stress field in DIC experiment can be obtained from the calculated strain field.

For example, the constitutive equation in **plane stress** case reads as (Voigt's notation):

$$\begin{bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \tau_{xy}\end{bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{bmatrix}\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \gamma_{xy}\end{bmatrix} \tag{5}$$

where E is Young's modulus, ν is Poisson's ratio, γ_{xy} is the engineering shear strain equals two times of infinitesimal shear strain ε_{xy}.

Similarly, the constitutive equation in **plane strain** case reads as (Voigt's notation):

$$\begin{bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \tau_{xy}\end{bmatrix} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\begin{bmatrix}1 & \frac{\nu}{1-\nu} & 0\\ \frac{\nu}{1-\nu} & 1 & 0\\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)}\end{bmatrix}\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \gamma_{xy}\end{bmatrix} \tag{6}$$

```
------------ Section 9 Start ------------
Material model to compute Cauchy stress fields:
    1: Linear elasticity -- Plane stress
    2: Linear elasticity -- Plane strain
    3: Others: User needs to edit codes in ./plotFiles/Plotstress.m
Input here: 1
-------------------------------------
Define Linear elasticity parameters
Young's modulus (unit: Pa):
Input here (e.g., 69e9): 69e9
Poisson's ratio:
Input here (e.g., 0.3): 0.3
-------------------------------------
Current image frame #: 2/2
------------ Section 9 Done ------------
```

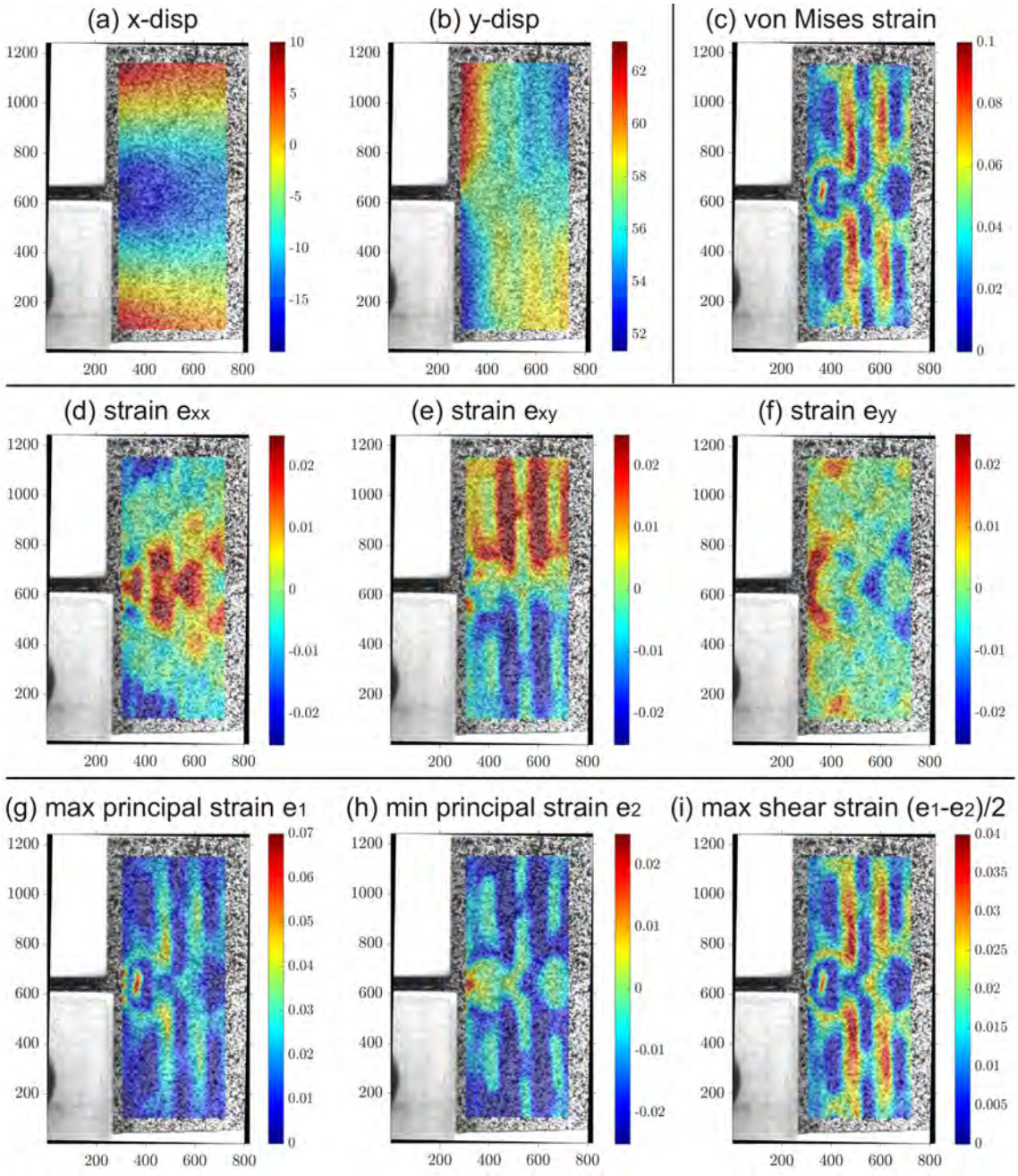


Figure 18: ALDIC tracked deformation fields with the original reference, undeformed image overlaid as the background image. (a-b) Solved x- and y-displacements (units: px). (c-i) Solved strain components: equivalent von Mises strain, e_{xx}, e_{xy}, e_{yy}, maximum principal strain, minimum principal strain, and maximum shear strain.

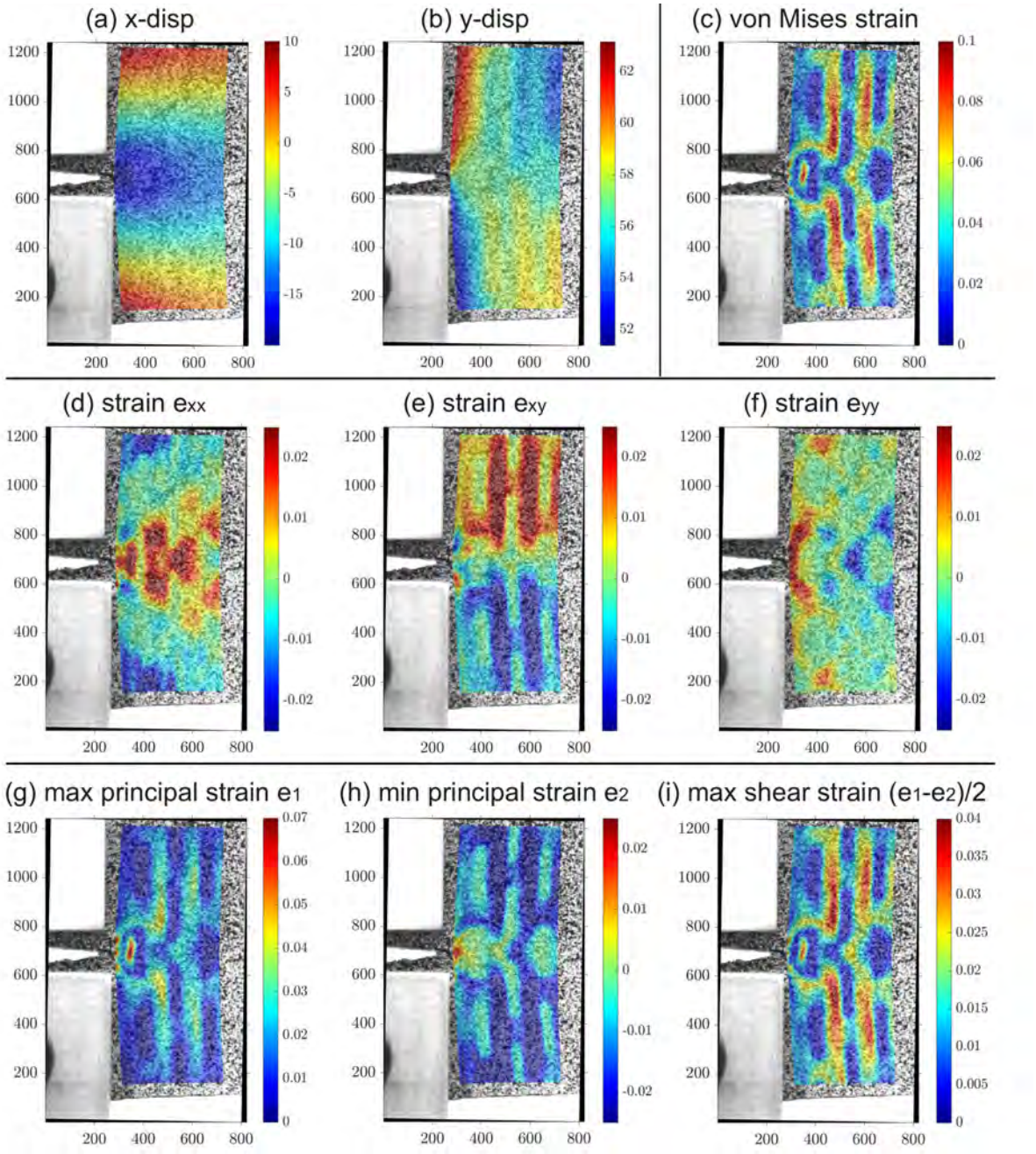


Figure 19: ALDIC tracked deformation fields with the current, deformed image overlaid as the background image. (a-b) Solved x- and y-displacements (units: px). (c-i) Solved strain components: equivalent von Mises strain, e_{xx}, e_{xy}, e_{yy}, maximum principal strain, minimum principal strain, and maximum shear strain.

12 Summary of the MATLAB command window screen outputs in the heterogeneous fracture example: "main_ALDIC.m"

This simulation is performed in MATLAB (R2018b, 64-bit) and on a workstation with an Intel i5-4670X CPU with a base clock of 3.40 GHz (4 threads), 16.0 GB memory, and runs under Windows

10.

12.1 Code Section 1

```
------------ Section 1 Start ------------
Building with 'MinGW64 Compiler (C++)'.
MEX completed successfully.
------------ Section 1 Done ------------
```

12.2 Code Section 2

```
------------ Section 2 Start ------------
Choose method to load images:
     0: Select images folder;
     1: Use prefix of image names;
     2: Manually select images.
Input here: 2
--- Please load first image ---
--- Please load next image ---
Do you want to load more deformed images? (0-Yes; 1-No)1

--- Define ROI corner points at the top-left and the bottom-right ---
Coordinates of top-left corner point are (309.500,69.500)
Coordinates of bottom-right corner point are (759.500,1145.500)

--- What is the subset size? ---
Each subset has an area of [-winsize/2:winsize/2, -winsize/2:winsize/2]
Input an even number: 20
--- What is the subset step? ---
Input an integer: 10

--- Method to solve ALDIC global step Subproblem 2 ---
    1: Finite difference(Recommended)
    2: Finite element method
Input here: 1

--- Set up Parallel pool ---
How many parallel pools to open? (Put in 1 if no parallel computing)
Input here: 4

--- Start to compute image gradients ---
--- Computing image gradients done ---
------------ Section 2 Done ------------
```

12.3 Code Section 3

```
Current image frame #: 2/2

------------ Section 3 Start ------------
```

```

--- Method to compute an initial guess of displacements ---
    0: Multigrid search based on an image pyramid
    1: Whole field search for all the subsets
    2: Search near manually clicked seeds and then interpolate for the
   full-field
Input here: 1

--- The size of initial guess search zone (pixels)? ---
User should start to try a small integer value, and gradually increase the
    value of
the search zone size until it is larger than the magnitudes of |disp u|
   and |disp v|.
User could also input [size_x, size_y] to search in a rectangular zone.
Input here: 60
Finish initial guess search!
--- Are you satisfied with initial guess with current search region? (0-
   yes; 1-no)? ---
Input here: 0
Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
Input here: 1
Do you clear bad points by directly clicking x-disp bad points? (0-yes; 1-
   no)
Input here: 1
Do you clear bad points by directly clicking y-disp bad points? (0-yes; 1-
   no)
Input here: 1
Finish setting up mesh and assigning initial value!
------------ Section 3 Done ------------
```

12.4 Code Section 4

```
------------ Section 4 Start ------------
***** Start step1 Subproblem1 *****
--- Set up Parallel pool ---
Local ICGN bad subsets %: 0/4494=0%
Elapsed time is 5.882090 seconds.
--- Start to manually remove bad points ---
Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
Input here: 1
Do you clear bad points by directly clicking x-disp bad points? (0-yes; 1-
   no)
Input here: 1
Do you clear bad points by directly clicking y-disp bad points? (0-yes; 1-
   no)
Input here: 1
--- Remove bad points done ---
------------ Section 4 Done ------------
```

12.5 Code Section 5

```
------------ Section 5 Start ------------
***** Start step1 Subproblem2 *****
Assemble finite difference operator D
Elapsed time is 0.377279 seconds.
Finish assembling finite difference operator D
Elapsed time is 0.542143 seconds.
------------ Section 5 Done ------------
```

12.6 Code Section 6

```
------------ Section 6 Start ------------
***** Start step2 Subproblem1 *****
Local ICGN bad subsets %: 0/4494=0%
Elapsed time is 5.139829 seconds.
***** Start step2 Subproblem2 *****
Elapsed time is 0.015518 seconds.
Update local step  = 0.021306
Update global step = 0.055187
**********************************

***** Start step3 Subproblem1 *****
Local ICGN bad subsets %: 0/4494=0%
Elapsed time is 4.518843 seconds.
***** Start step3 Subproblem2 *****
Elapsed time is 0.012501 seconds.
Update local step  = 0.015562
Update global step = 0.014917
**********************************

***** Start step4 Subproblem1 *****
Local ICGN bad subsets %: 0/4494=0%
Elapsed time is 9.440746 seconds.
***** Start step4 Subproblem2 *****
Elapsed time is 0.012753 seconds.
Update local step  = 0.010151
Update global step = 0.0038372
**********************************

***** Start step5 Subproblem1 *****
Local ICGN bad subsets %: 0/4494=0%
Elapsed time is 5.592309 seconds.
***** Start step5 Subproblem2 *****
Elapsed time is 0.015665 seconds.
Update local step  = 0.0077425
Update global step = 0.0026579
**********************************

------------ Section 6 Done ------------
```

12.7 Code Section 7

```
------------ Section 7 Start ------------
***** Check convergence *****
==== uhat^(k) - u^(k) ====
0.0062234
0.054471
0.044709
0.042504
0.042029
==== Fhat^(k) - F^(k) ====
0.003901
0.002719
0.00050623
0.00014638
9.7637e-05
==== uhat^(k) - uhat^(k-1) ====
0.055187
0.014917
0.0038372
0.0026579
==== udual^(k) - udual^(k-1) ====
0.0018965
0.00047513
0.00013772
9.1862e-05
==== vdual^(k) - vdual^(k-1) ====
0.018756
0.0060879
0.0036196
0.0027162
------------ Section 7 Done ------------
```

12.8 Code Section 8

```
------------ Section 8 Start ------------
Do you want to smooth displacement? (0-yes; 1-no)
Input here: 1
-------------------------------------
What method to use to compute strain?
    0: Direct output from ALDIC;
    1: Finite difference(Recommended);
    2: Plane fitting;
    3: Finite element;
Input here: 1
-------------------------------------
Infinitesimal strain or finite strain?
    0: Infinitesimal strain;
    1: Eulerian strain;
    2: Green-Lagrangian strain;
    3: Others: code by yourself;
Input here: 0
-------------------------------------
```

```
Over which image(s) you want to plot the results?
    0: First image only (if you are using only two images);
    1: Second and next images;
Input here: 1
-------------------------------------
Save figures into the format:
    1: jpg(Choose transparency 0~1)
    2: pdf(Choose transparency = 1)
    3: Others: Edit codes in ./plotFiles/SaveFigFiles.m
Input here: 1
-------------------------------------
Define transparency for overlaying original images:
Input a real number between 0(Only original images)
and 0.99 (Non-transparent deformation results).
Input here(e.g., 0.5): .5
-------------------------------------
Current image frame #: 2/2
------------ Section 8 Done ------------
```

12.9 Code Section 9

```
------------ Section 9 Start ------------
Material model to compute Cauchy stress fields:
    1: Linear elasticity -- Plane stress
    2: Linear elasticity -- Plane strain
    3: Others: User needs to edit codes in ./plotFiles/Plotstress.m
Input here: 1
-------------------------------------
Define Linear elasticity parameters
Young's modulus (unit: Pa):
Input here (e.g., 69e9): 69e9
Poisson's ratio:
Input here (e.g., 0.3): 0.3
-------------------------------------
Current image frame #: 2/2
------------ Section 9 Done ------------
```

13 Stress concentration in plate with a circular hole: "main_ALDIC_QuadtreeHole.m"

In practice, the DIC test sample may have a complex shape, and a uniform rectangular Q4 mesh does not work well near complex geometry. For example, ALDIC uses a uniform Q4 mesh to track the deformation field in the 2D plane tensile test is inaccurate near the central circular hole, as shown in Fig. 23.

There are already several strategic solutions to improve these inaccuracies [21]. For example, a practical strategy is to remove these bad points where there are not enough speckles. Another strategy is to move the control point of a bad local subset to the a region where a good pattern is painted, or to apply other subset-splitting techniques.

Here we provide another efficient strategy to solve the complex geometry, where all the DIC computation are performed on an adaptive quadtree mesh (cf Fig. 20(c)) which is manually defined (Section 13) or automatically generated from the original image or a binary image mask file (see next section: Section 14).

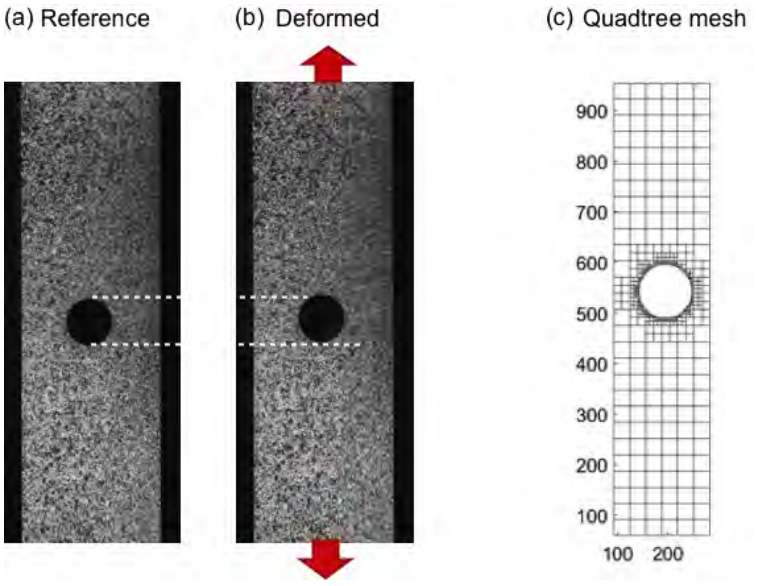


Figure 20: (a-b) Reference & deformed DIC raw images, and (c) generated quadtree mesh of a 2D plane tensile test with a circular hole.

To test this section, the user should open the main file "main_ALDIC_QuadtreeHole.m" instead of "main_ALDIC.m". In the "main_ALDIC_QuadtreeHole.m" code Section 2, the user should upload an image folder, for example, `Images_ForQuadtree_Sample12` instead of previous `Images_frac_heter`, see Fig. 21. Then the user could manually define the geometry of the circular hole by clicking several random points along the hole boundary, see Fig. 22, and a Taubin fitting process [22] is used to fit the circular hole shape.

With manually defined circular hole geometry, an adaptive quadtree mesh will be generated, as shown in Fig. 20(c). All pixels in the hole will not be used during the following procedure. Finally,

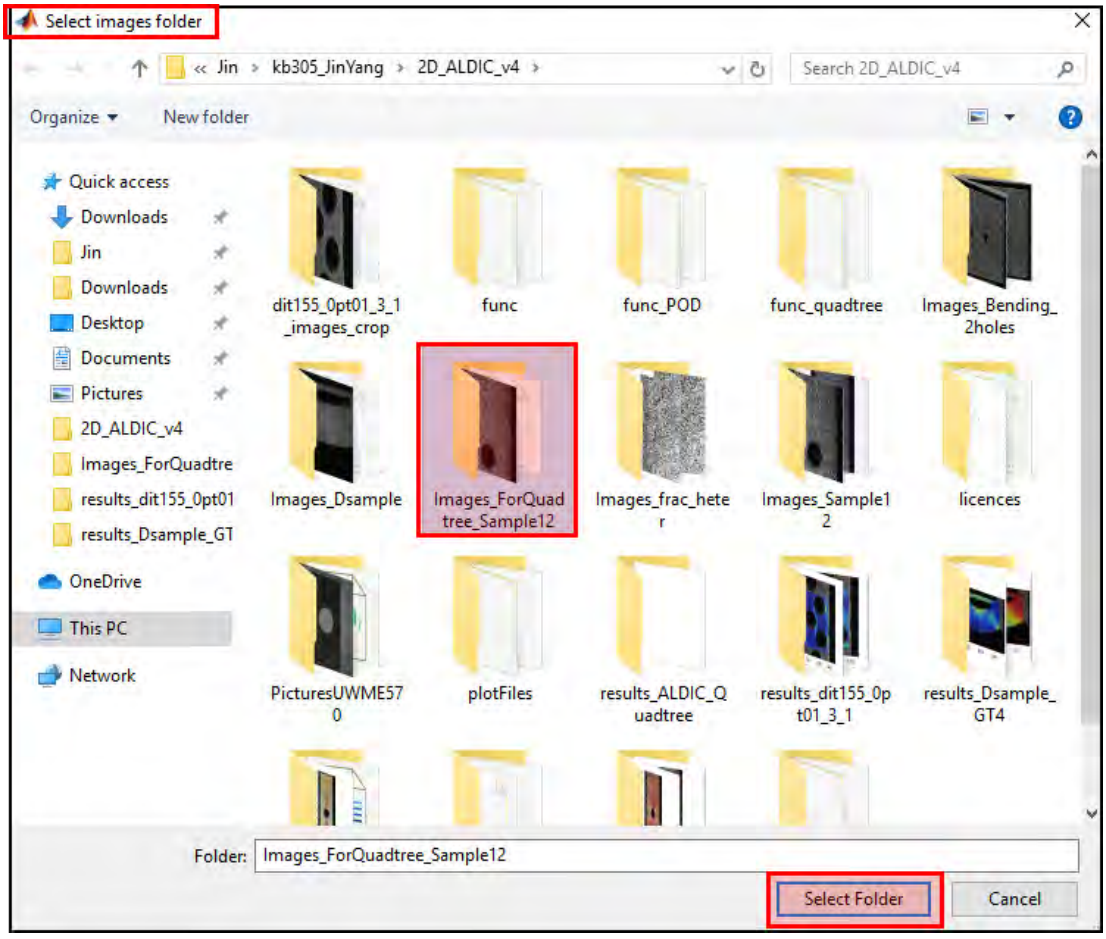


Figure 21: Image folder "Images_ForQuadtree_Sample12" is selected in the "main_ALDIC_QuadtreeHole.m" example.

the ALDIC code uses generated quadtree mesh and tracks unknown deformation field of a 2D plane tensile test with a circular hole. All the results are summarized in Fig. 24.

14 Automatically generated quadtree mesh from DIC raw images: "main_ALDIC_Quadtree.m"

Quadtree meshes could automatically be generated from the original reference image or a binary image mask file. To test this section, the user should open the main file "main_ALDIC_Quadtree.m" instead of "main_ALDIC.m". In the "main_ALDIC_Quadtree.m" code Section 2, the user should also upload an image folder, for example, `Images_ForQuadtree_Sample12` instead of previous `Images_frac_heter` , as shown in Fig. 21.

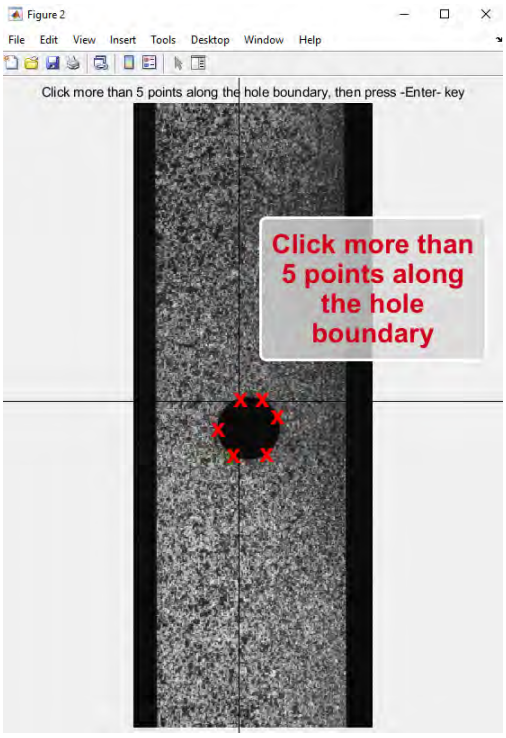


Figure 22: Users could define the circular hole geometry by manually clicking more than five points along the hole boundary.

14.1 Select the reference frame or an image masking file

File “main_ALDIC_Quadtree.m” is almost identical to the previous file “main_ALDIC_QuadtreeHole.m”. In code Section 2, the user will be asked to select an image frame to compute the image mask file. The user could also upload a binary image mask file directly, see Fig. 25.

The user will then be asked to randomly click a few points in the image where no speckles are painted. After that the user will also be asked to randomly click a few points on the sample surface where a speckle pattern is painted.

After eliminating noise or small dots/blobs smaller than the input removal radius, a binary image mask file, as shown in Fig. 27(c) is obtained.

With the obtained binary image mask file, the adaptive quadtree mesh can be generated automatically, and all pixels in the background are not used in the following procedures.

All relevant MATLAB command window screen outputs are summarized below.

```
--- Define image mask file step 1: background ---
Click more than 6 points in the background,
then press the -Enter- key.

--- Define image mask file step 2: sample surface ---
Click more than 6 points on the sample pattern,
```

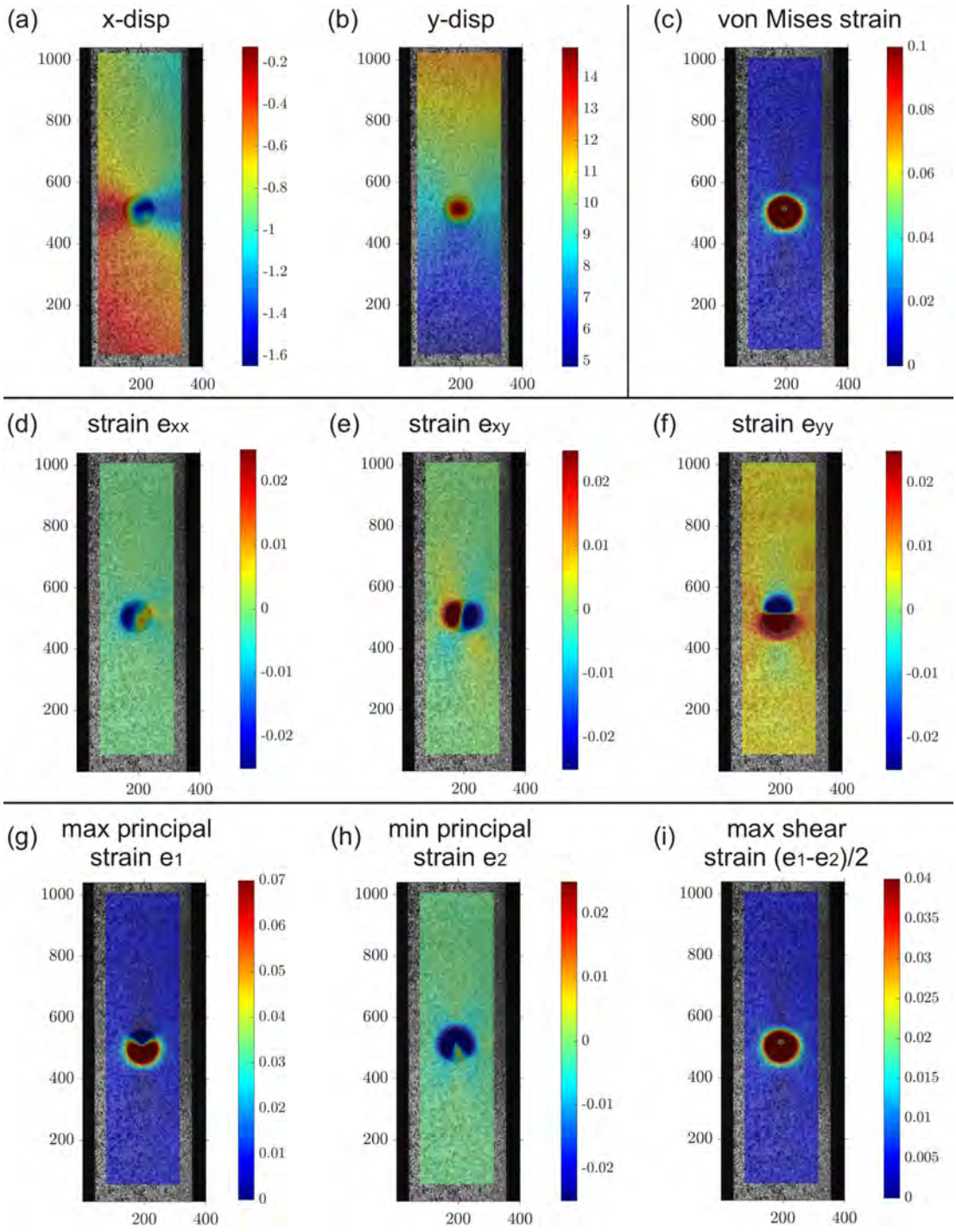


Figure 23: ALDIC uses a uniform Q4 mesh to track the deformation field in the 2D plane tensile test is inaccurate near the central circular hole. (a-b) Solved x- and y-displacements (units: px). (c-i) Solved strain components: equivalent von Mises strain, e_{xx}, e_{xy}, e_{yy}, maximum principal strain, minimum principal strain, and maximum shear strain.

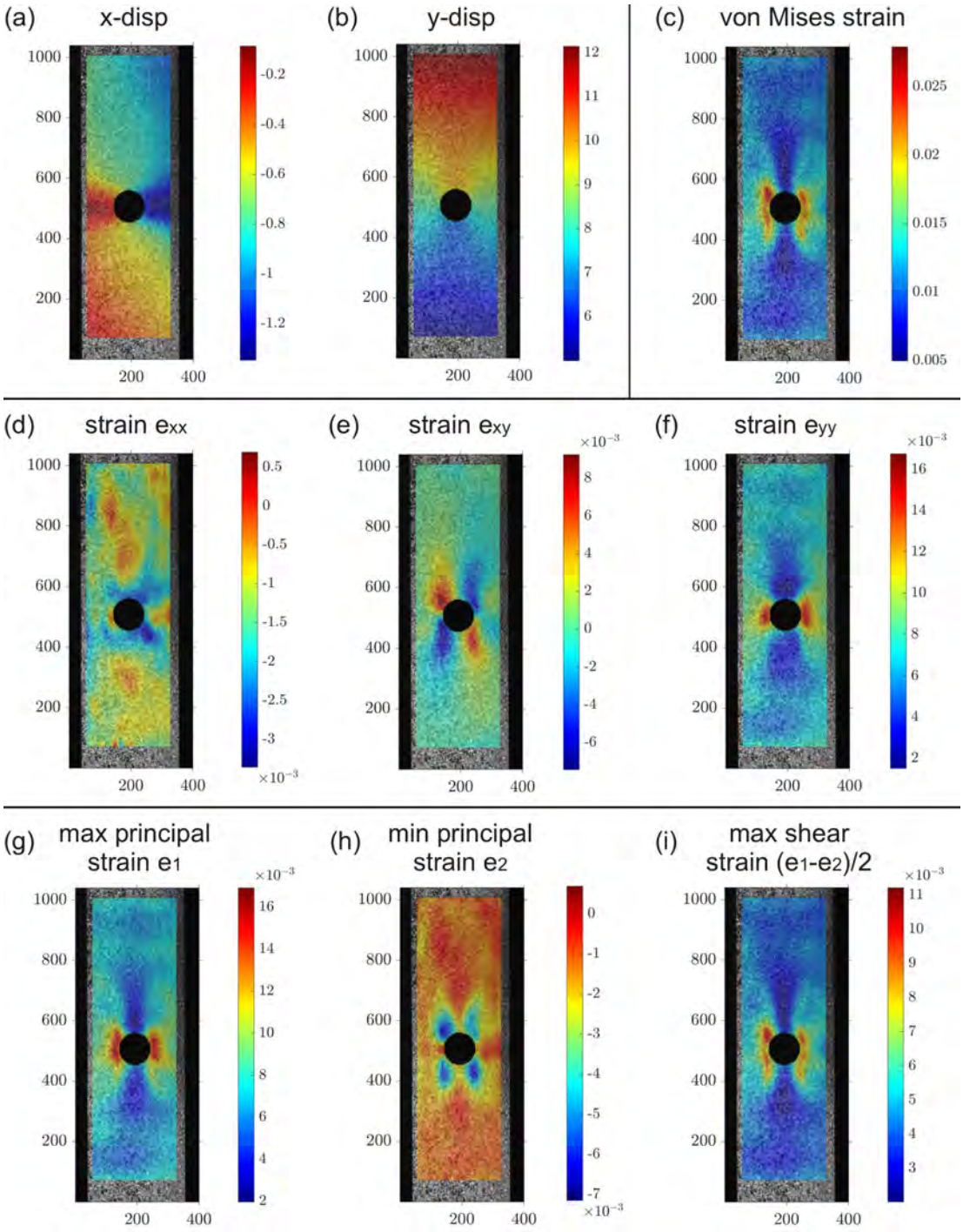


Figure 24: ALDIC uses a quadtree mesh to track the deformation field in the 2D plane tensile test is accurate near the central circular hole. (a-b) Solved x- and y-displacements (units: px). (c-i) Solved strain components: equivalent von Mises strain, e_{xx}, e_{xy}, e_{yy}, maximum principal strain, minimum principal strain, and maximum shear strain.

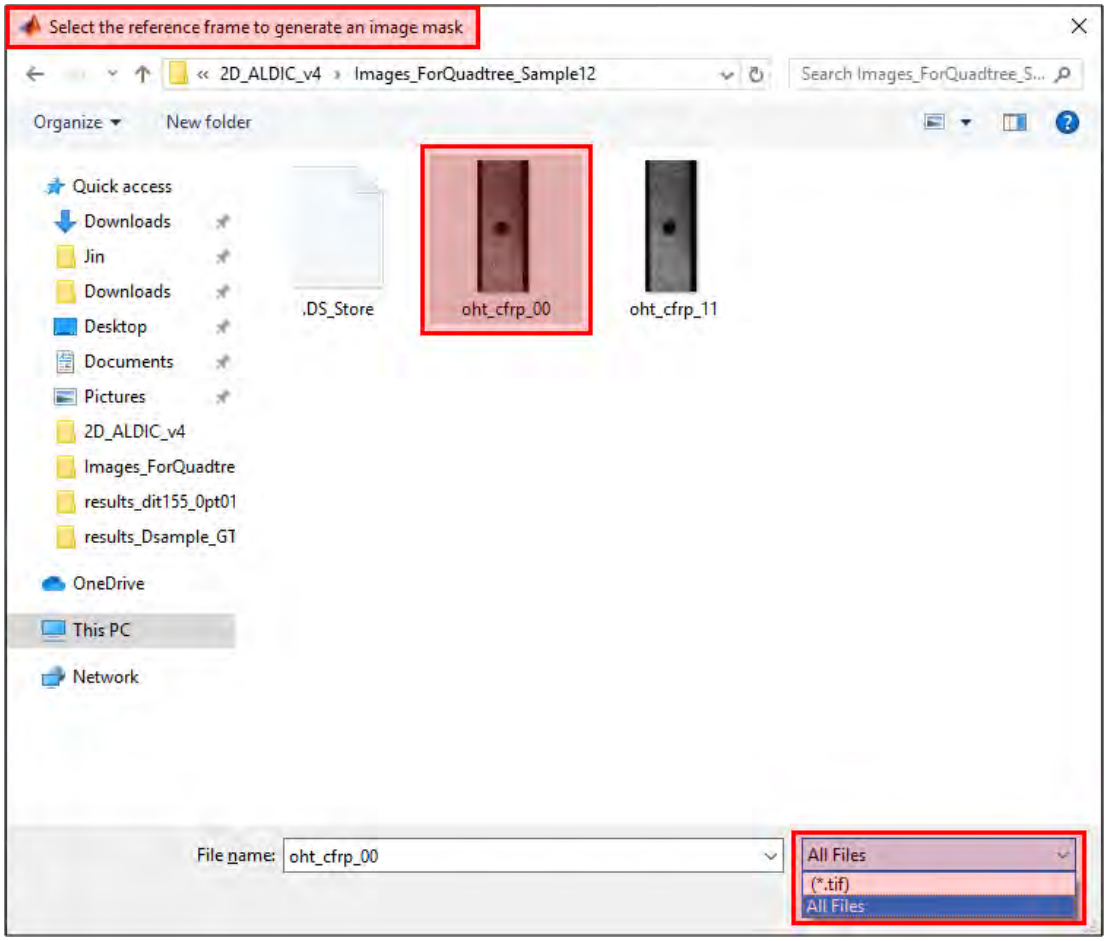


Figure 25: Here the first reference image is selected and used to calculate a binary image mask file. The user can also upload a binary image mask file directly.

```
then press the -Enter- key.

Is the background darker or brighter than the sample surface?
    0: Background is darker;
    1: Background is brighter;
Input here: 0

Remove all objects smaller than the critical radius.
Input here: 2
Are you satisfied with current removal radius? (0-Yes; 1-No)
Input here: 1

Remove all objects smaller than the critical radius.
Input here: 5
Are you satisfied with current removal radius? (0-Yes; 1-No)
Input here: 1

```

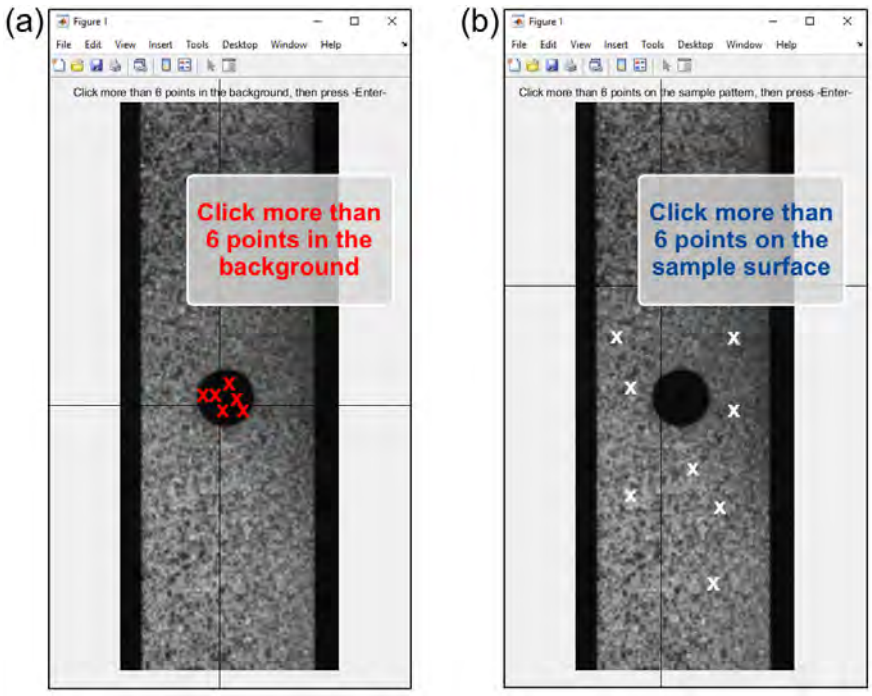


Figure 26: The user will be asked to randomly click several points in the image where no speckles are painted, and then to randomly click several points on the sample surface with a speckle pattern.

```
Remove all objects smaller than the critical radius.
Input here: 10
Are you satisfied with current removal radius? (0-Yes; 1-No)
Input here: 0

--- Start to compute image gradients ---
--- Computing image gradients done ---
------------ Section 2 Done ------------
```

Here I also provide two other examples where the ALDIC code can solve unknown deformations of tested samples with complex geometry, see Figs. 28-31 [23, 24].

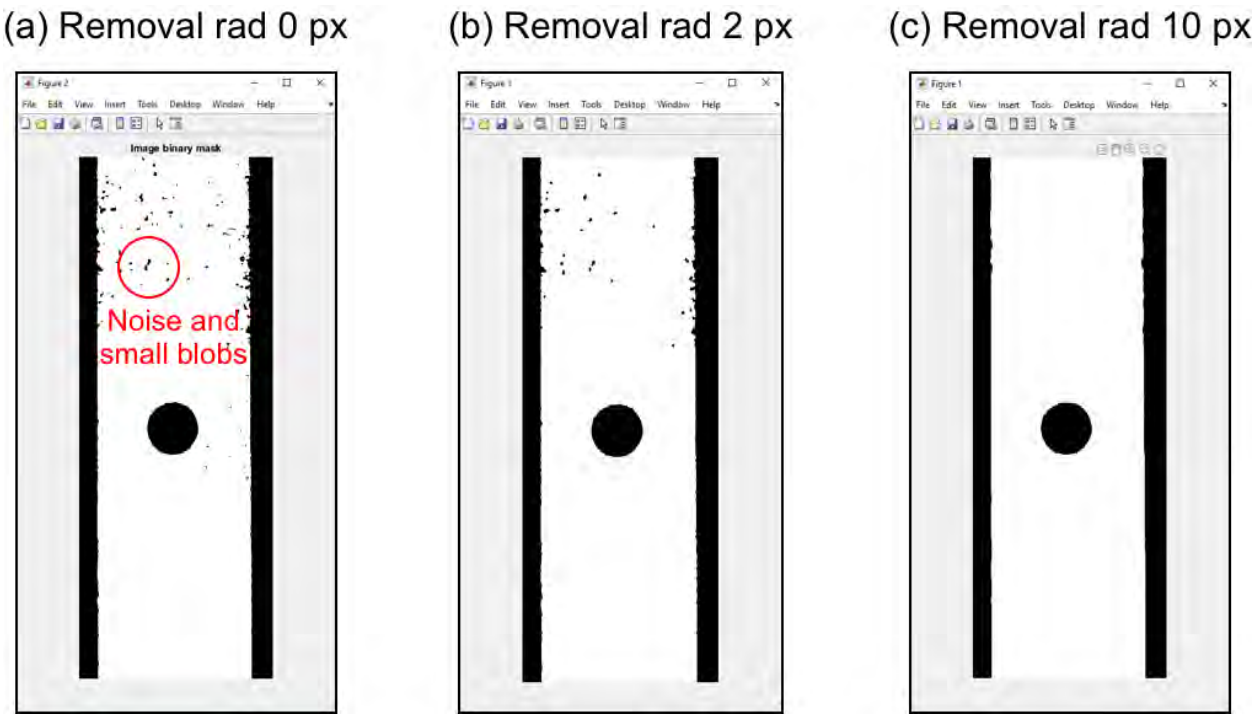


Figure 27: After eliminating noise or small blobs that are smaller than the input removal radius, a binary image mask file (c) is obtained.

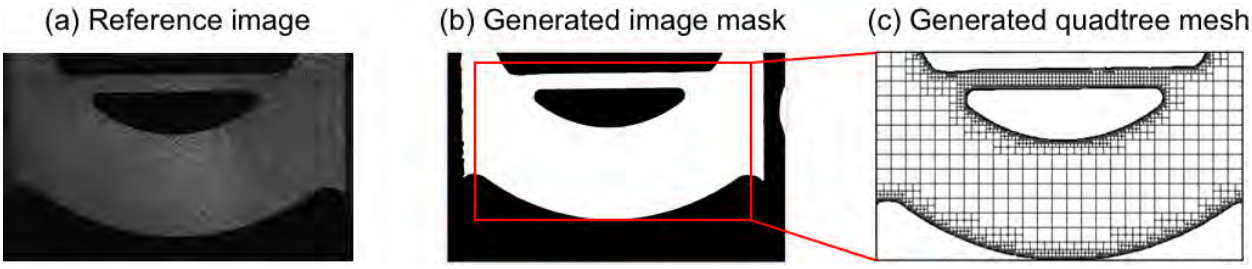


Figure 28: Generated image mask file and quadtree mesh of a D-shape specimen.

15 Frequently Asked Questions (FAQs)

15.1 About MATLAB mex set up

15.1.1 Where do I install a TDM-gcc compiler?

This is a general question for all MATLAB users to apply mex C/C++ compilers where more details can be found online, such as [14, 15]. Users need to install suitable mex C/C++ compilers based on their OS and MATLAB versions. In my experience, the mex C/C++ compiler usually is already installed on MAC system. For Windows users, I usually install the TDM-gcc compiler on a first level main disk, such as “C:, D:, E:”, and higher level disks (like “C/FirstLevel/SecondLevel/...”) may not

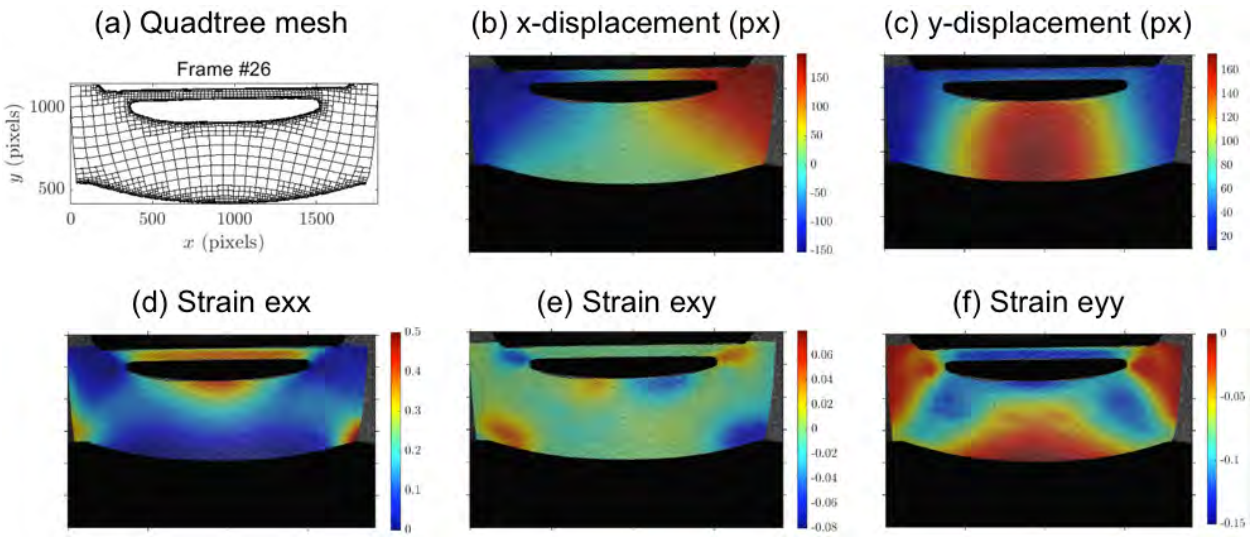


Figure 29: ALDIC tracked deformation field of a tensile test of a D-shape specimen.

work. If you have better suggestions, I appreciate you could let me know.

15.1.2 Mex permission denied

MATLAB command window reports an error message when mex permission is denied, see Fig. 32. This error appears because a mex has already been set. To fix this bug, please update the first code section `mex -O ba_interp2.cpp;` to `try mex -O ba_interp2.cpp; catch; end` .

15.1.3 File "ba_interp2.cpp" is not found

MATLAB command window reports an error message when the "ba_interp2.cpp" file is not found, see Fig. 33. This error happens because "ba_interp2.cpp" file does not exist in MATLAB "Current Folder". To fix this bug, please enter the "2D_ALDIC_v$*$" folder and then re-run the "main_ALDIC.m" or
"main_ALDIC_Quadtree.m" code Section 1. Alternatively, the user could also copy and paste a "ba_interp2.cpp" file to your current folder.

15.1.4 What if I want to use the MATLAB default interpolation instead of "ba_interp2"?

You can still use the ALDIC MATLAB code! You need to convert the "ba_interp2" to other MATLAB interpolation functions. This change needs to be done in the "./func/funICGN.m" and "./func/-funICGN_Subpb1.m" to change the "interpmethod". For example, you can replace the function "ba_interp2" with "interp2", where interp2 is the MATLAB built interpolation function. You can also try other interpolation schemes too. In my experience, "ba_interp2" runs much faster than the

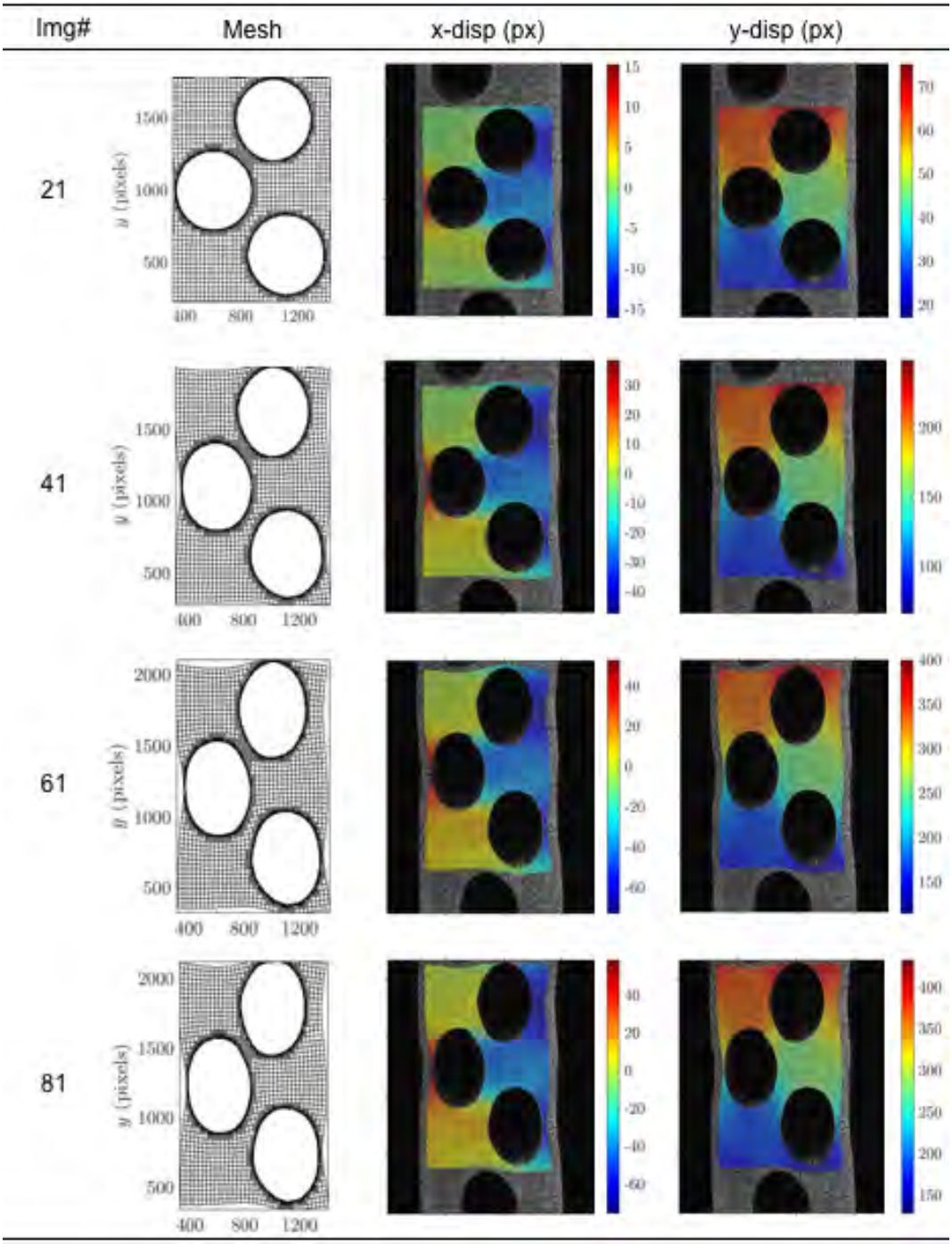


Figure 30: ALDIC tracked displacement fields of a tensile test of an open-cell elastomeric foam.

MATLAB default built cubic interpolation script. So if “ba_interp2” can be used, that will be a better choice.

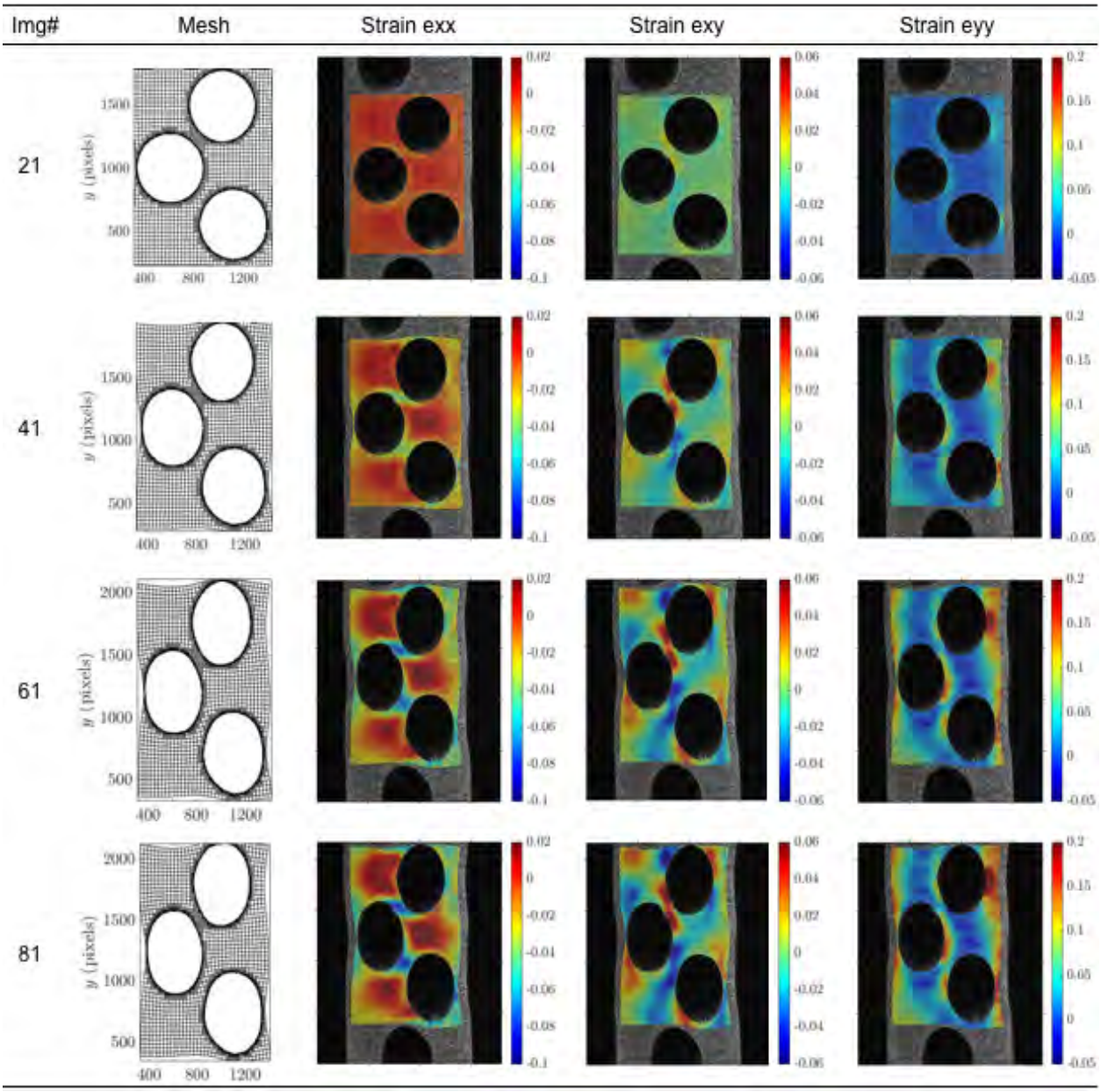


Figure 31: ALDIC tracked infinitesimal strain fields of a tensile test of an open-cell elastomeric foam.

15.2 About MATLAB parallel computing

15.2.1 How to set up parallel computing preference?

First, the user needs to make sure he/she has already installed the parallel computing toolbox. Then, all parallel computing toolbox preferences can be modified by the user as shown in Fig. 7. The maximum number of workers in a parallel pool is also limited by the computer hardware

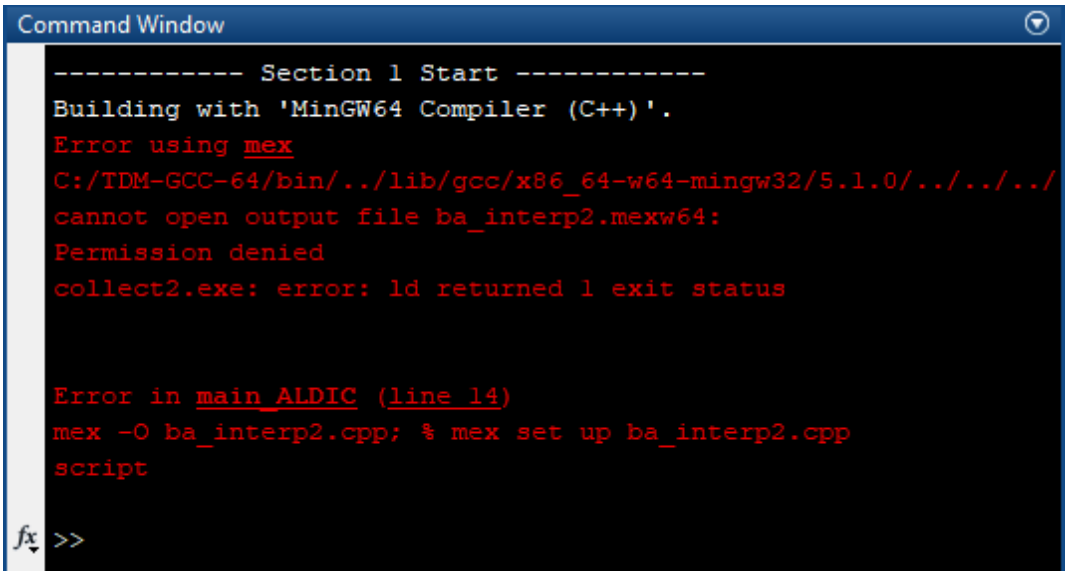


Figure 32: MATLAB command window reports an error message when mex permission is denied.

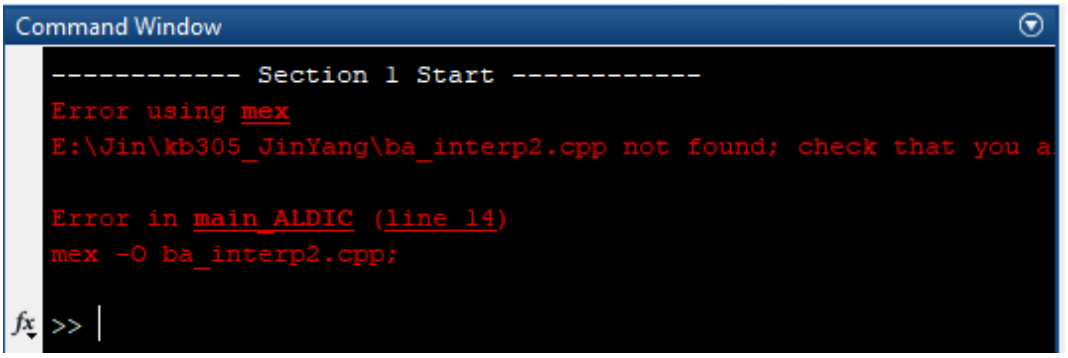


Figure 33: MATLAB command window reports an error message when the "ba_interp2.cpp" file can not be found.

condition.

15.2.2 What if I don't have a parallel computing toolbox?

The user can easily install the parallel computing toolbox. If you don't want to apply parallel computing, ALDIC still runs very fast, and the parameter `DICpara.ClusterNo` needs to be set as `0` or `1` .

15.3 About ALDIC algorithm

15.3.1 MATLAB reports error "Undefined function or variable"

Please make sure you have already added all the subfolders ('./src','./plotfiles', ...) and the DIC image folder on the MATLAB path.

15.3.2 Which image to load as the first image?

One comment about the ALDIC code is that it is set by default to manipulate the deformed images and attempt to transform deformed images back to the reference image and compute their deformation fields, which is based on the *Lagrangian* description (see more details in Appendix 15.3.5). If the user wants to track the deformation field in the *Eulerian* description, he/she can select the reference image as the second image, and select the deformed image as the first image and manipulate the reference image to transform to the current deformed image.

15.3.3 Error "Insufficient data for surface estimation" when using "gridfit"

This problem usually happens when there are few local subsets get convergence in Subproblem 1 (local step), which breaks down the `gridfit` function [25]. The user can check the bad subsets percentage which is expected to display on the command window. For example, if there are almost 100% bad subsets, which means all subsets are diverged in the ALDIC Subproblem 1.

Here are three possible steps to fix this problem: (i) In code Section 2, check DIC parameters: subset size (`DICpara.winsize`). In 2D-DIC, each subset is expected to have at least $3 \sim 5$ features (e.g., speckle dots) in your DIC pattern. Usually a larger subset size help improve the convergence of the ALDIC ADMM Subproblem 1. But the subset size cannot be too large since it will also decrease the DIC overall spatial resolution. In theory, distance between neighboring subsets can be an arbitrary integer. Considering speed and accuracy, I recommend the window step size (`DICpara.winstepsize`) is set to be $(0.25 \sim 1)$ times of the window size (`DICpara.winsize`).

(ii) In code Section 3, check the initial guess, and plot initial guess `U0` to see whether the initial guess of the unkown deformation field makes sense:

```
U0 = Init(u,v,cc.max,DICmesh.x0,DICmesh.y0,0); PlotuvInit;
```

The "whole field FFT search" is the most robust method compared with the multiscale search, where the user can manually define the size of the search zone.

(iii) After checking (i-ii), please re-run code Section 4 and see whether each local subset obtains convergence or not. The user can also display this info message on the command window by uncommenting a line in "./src/LocalICGN.m" starting from "display(...);". If there are still problems or most subsets still dont't get converged. Feel free to report the error message or send a pair of your DIC images to: aldicdvc@gmail.com.

15.3.4 How to compute other types of strains?

(i) What the ALDIC code solved are displacements ($\mathbf{U} = \mathbf{x}(\mathbf{X}) - \mathbf{X}$) and deformation gradients ($\mathbf{F} = \nabla_{\mathbf{X}}\mathbf{U} = \nabla_{\mathbf{X}}\mathbf{x} - \mathbf{I}$), where $\mathbf{x}$ and $\mathbf{X}$ are the coordinates in deformed and reference configurations, respectively. $\mathbf{U}$ is stored in `ResultDisp`, and $\mathbf{F}$ is stored in `ResultDefGrad`.

In ALDIC, length of the displacement vector – $\mathbf{U}$ – is two times of the row number of the FE-mesh coordinates (ResultFEMesh{1}.coordinatesFEM), and $\mathbf{U}$ vector is assembled by the nodal displacements $[u^1, v^1, u^2, v^2, \cdots, u^N, v^N]^T$ with nodal indices as $[1, 2, \cdots, N]$. The length of $\mathbf{F}$ vector is four times of the row number of FE-mesh coordinates and is assembled in the order of $[F_{11}^1, F_{21}^1, F_{12}^1, F_{22}^1, F_{11}^2, F_{21}^2, F_{12}^2, F_{22}^2, \cdots, F_{11}^N, F_{21}^N, F_{12}^N, F_{22}^N]^T$.

(ii) With the ALDIC solved displacements $\mathbf{U}$ and deformation gradients $\mathbf{F}$, we calculate strains in code Section 8. For example, if strains are small ($< 5\%$), infinitesimal strain is a good option, where $e_{xx} = F_{11}$, $e_{yy} = F_{22}$, $e_{xy} = 0.5\,(F_{12}+F_{21})$. For engineering shear strains, $E_{xx} = F_{11}$, $E_{yy} = F_{22}$, $E_{xy} = (F_{12} + F_{21})$. With computed $\mathbf{F}$ or `FStraintemp`, other types of finite strain measurements can be easily computed based on user's choice.

(iii) Comment about difference between $\mathbf{F}$ and `FStraintemp`. In code Section 8, you will be also be asked:

```
What method to use to compute strain?
0: Direct output from ALDIC;
1: Finite difference(Recommended);
2: Plane fitting;
3: Finite element;
```

If you choose the method 0, `FStraintemp` is exactly the same with computed $\mathbf{F}$; If you choose the method 1, method 2, or method 3, the `FStraintemp` is re-computed from the computed u. The size of `FStraintemp` is cropped by `Rad`, since the strains near the edges are less accurate. The coordinates of `FStraintemp` are: `x0(1+Rad:M-Rad,1+Rad:N-Rad), y0(1+Rad:M-Rad,1+Rad:N-Rad)`.

15.3.5 Image vs. physical world coordinate systems and Lagrangian vs. Eulerian description

During the DIC post-processing and visualization, the user should pay attention to the used coordinate system and the Lagrangian/Eulerian description, as shown in Fig. 34. For example, going down is the positive direction in the image coordinate system while it is the negative direction in the physical world coordinate system. Different coordinate systems will give different vertical displacements and shear strain results. Specifically in the ALDIC code, we name all the variables in the physical world coordinate system ended with the text " `World` ".

In addition, I also show the difference between the Lagrangian description vs. the Eulerian description as shown in Fig. 34. For example, in the Lagrangian description, for the same "particle" (x_1, x_2), we are always measuring its displacement and strain at different time points. However,

in the Eulerian description, the same "position" (y_1, y_2) can be different particles at different time points. It is generally more common to use Lagrangian approach in solid mechanics and use Eulerian approach to fluid flows. But this is not always the case. The user should know their difference and know which desciption they use.

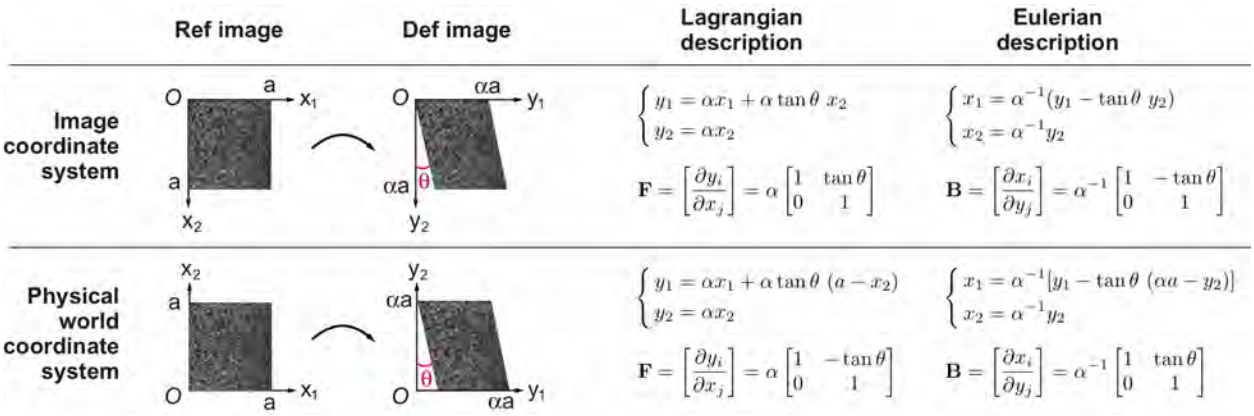


Figure 34: (i) Comparison between the image coordinate system vs. physical world coordinate system. (ii) Comparison between the Lagrangian description and the Eulerian description.

Acknowledgements

I am grateful to Dr. Louisa Avellar for sharing her unpublished images of fracture. I want to thank Prof. Christian Franck, Prof. Can C. Aydıner, Dr. Alex K. Landauer, Jialiang Tao, Zhan Ma, many students who took course ME570 at the University of Wisconsin-Madison, and all other ALDIC users for lots of helpful feedback and discussions. I also want to thank Prof. Jose C. Outeiro for helpful suggestions to improve the ALDIC code to be used in the machining manufacturing processes. I also acknowledge the support of the US Air Force Office of Scientific Research through the MURI grant 'Managing the Mosaic of Microstructure' (FA9550-12-1-0458).

References

[1] J Yang and K Bhattacharya. Augmented Lagrangian Digital Image Correlation. *Experimental Mechanics*, 59:187–205, 2019.

[2] *2D ALDIC code*. https://www.mathworks.com/matlabcentral/fileexchange/70499-augmented-lagrangian-digital-image-correlation-and-tracking.

[3] S Boyd, N Parikh, E Chu, B Peleato, and J Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Machine Learning*, 3:1–122, 2010.

[4] *FE-Global-DIC code*. https://www.mathworks.com/matlabcentral/fileexchange/82873-2d-finite-element-global-digital-image-correlation-fe-dic.

[5] https://www.researchgate.net/publication/329456141_Augmented_Lagrangian_Digital_Image_Correlation.

[6] J Yang and K Bhattacharya. Combining image compression with digital image correlation. *Experimental Mechanics*, 59:629–642, 2019.

[7] J Yang and K Bhattacharya. Fast adaptive global digital image correlation. In L Lamberti, MT Lin, C Furlong, C Sciammarella, PL. Reu, and MA Sutton, editors, *Advancement of Optical Methods & Digital Image Correlation in Experimental Mechanics, Volume 3*, pages 69–73. Springer, 2019.

[8] J Yang and K Bhattacharya. Fast Adaptive Mesh Augmented Lagrangian Digital Image Correlation. *Experimental Mechanics*, 61:719–735, 2021.

[9] J Yang, L Hazlett, A.K. Landauer, and C. Franck. Augmented Lagrangian Digital Volume Correlation. *Experimental Mechanics*, 2020.

[10] https://www.researchgate.net/publication/343676441_Augmented_Lagrangian_Digital_Volume_Correlation.

[11] *ALDVC code*. https://www.mathworks.com/matlabcentral/fileexchange/77019-augmented-lagrangian-digital-volume-correlation-aldvc.

[12] Augmented Lagrangian Digital Volume Correlation (ALDVC) Code Manual.

[13] *2D Image Interpolation with ba_interp2*. https://www.mathworks.com/matlabcentral/fileexchange/20342-image-interpolation-ba_interp2.

[14] *MATLAB Support for MinGW-w64 C/C++ Compiler*. https://www.mathworks.com/matlabcentral/fileexchange/52848-matlab-support-for-mingw-w64-c-c-compiler.

[15] *MathWorks: MinGW-w64 Compiler*. https://www.mathworks.com/help/matlab/matlab_external/install-mingw-support-package.html.

[16] *MathWorks Help Center: parpool*. https://www.mathworks.com/help/distcomp/parpool.html.

[17] E Bar-Kochba, J Toyjanova, E Andrews, K-S Kim, and C Franck. A fast iterative digital volume correlation algorithm for large deformations. *Experimental Mechanics*, 55:261–274, 2015.

[18] AK Landauer, M Patel, DL Henann, and C Franck. A q-factor-based digital image correlation algorithm (qDIC) for resolving finite deformations with degenerate speckle patterns. *Experimental Mechanics*, 58:815–830, 2018.

[19] *FIDVC code*. https://github.com/FranckLab/FIDVC.

[20] *qFIDVC code*. https://github.com/FranckLab/qFIDVC.

[21] J Blaber, B Adair, and A Antoniou. Ncorr: open-source 2d digital image correlation matlab software. *Experimental Mechanics*, 55(6):1105–1122, 2015.

[22] G Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (11):1115–1138, 1991.

[23] *DIC Challenge 3D-DIC Sample 3 (*_0 images, D-Specimen tensile test)*. https://sem.org/3ddic.

[24] AK Landauer, XQ Li, C Franck, and Henann D. Experimental characterization and hyperelastic constitutive modeling of open-cell elastomeric foams. *Journal of the Mechanics and Physics of Solids*, (133):103701, 2019.

[25] *Gridfit code*. https://www.mathworks.com/matlabcentral/fileexchange/8998-surface-fitting-using-gridfit.