Augmented Lagrangian Digital Volume Correlation (ALDVC) Code Manual (v1.0.3)

Jin Yang[1,†], Lauren Hazlett[1,2,3], Alex Landauer[1,2], Christian Franck[1,‡]

[1] Department of Mechanical Engineering, University of Wisconsin-Madison, Madison, WI, USA
[2] School of Engineering, Brown University, Providence, RI, USA
[3] Center for Biomedical Engineering, Brown University, Providence, RI, USA

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

Github page: https://github.com/FranckLab/ALDVC
MATLAB FileExchange page: https://www.mathworks.com/matlabcentral/fileexchange/77019-augmented-lagrangian-digital-volume-correlation-aldvc

Last updated on September 17, 2020

Contents

1 Introduction

Digital volume correlation (DVC), the volumetric extension of the popular digital image correlation (DIC) technique, is a powerful experimental tool for measuring 3D volumetric full-field displacements and strains. Most current DVC algorithms can be categorized into either local or finite-element-based global methods. As with most experimental approaches, there are drawbacks with each of these methods. In the local method the subvolume 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 volumetric 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 volumetric data sets. Here we present a new hybrid DVC algorithm, called augmented Lagrangian digital volume correlation (ALDVC) [1], which combines the advantages of both the local (fast computation times) and global (compatible displacement field) methods. This new algorithm builds on our recent work on the augmented Lagrangian digital image correlation (2D-ALDIC) technique [2] (MATLAB 2D-ALDIC code link: [3]) and solves the general motion optimization problem by using the alternating direction method of multipliers (ADMM) [4]. We demonstrated [1] that our ALDVC algorithm has high accuracy and precision while maintaining low computational cost, and is a significant improvement compared to current local and global DVC methods.

For a review of both local and global DVC methods, and details of this new proposed ALDVC method, please see Fig. 1 and our paper [1] (full text can also be requested at [5]).

Some advantages of our ALDVC algorithm are highlighted below:

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

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

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

(iv) Since global compatibility is enforced the user does not need to choose a specific displacement smoothing filters.

(v) It works well with compressed images and can include adaptive mesh refinement [6].

(vi) It can solve an image sequence with multiple time frames [1].

2 Code installation

The ALDVC code can be downloaded at [3]. It has been tested on MATLAB versions later than R2018a for Windows 10. The Parallel Computing Toolbox is highly recommended (not required) to speed up the code.

[1] Cumulative mode is already implemented; incremental mode will be added soon. Email us if you want more information: aldicdvc@gmail.com -or- jyang526@wisc.edu

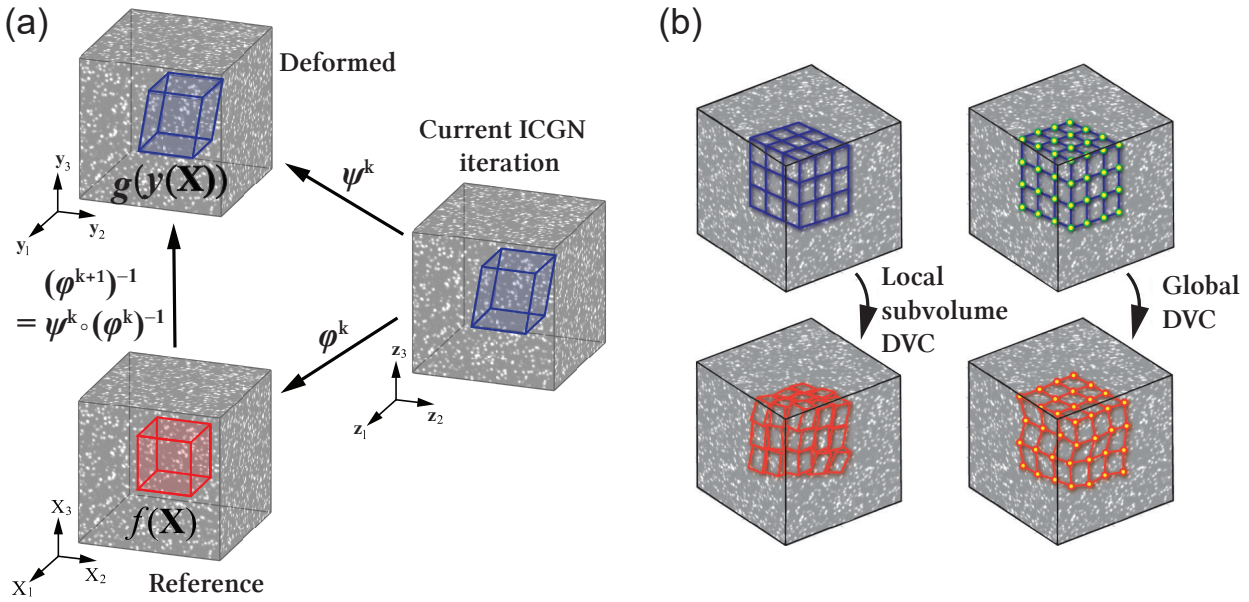


Figure 1: (a) Schematic showing a volumetric DVC reference image f(**X**), with a general speckle pattern, deforming into the deformed image g(**y**(**X**)) under some mapping **y** and the change of variables involved within the IC-GN iteration in the local subvolume DVC method. **X** and **y** coordinates are in the reference and deformed images, respectively. **z** coordinates are in current IC-GN iteration. (b) A schematic comparison between the local DVC method (left), where all the subvolumes are analyzed independently, and the global DVC method (right), where a global basis set is used to represent the full-field deformation.

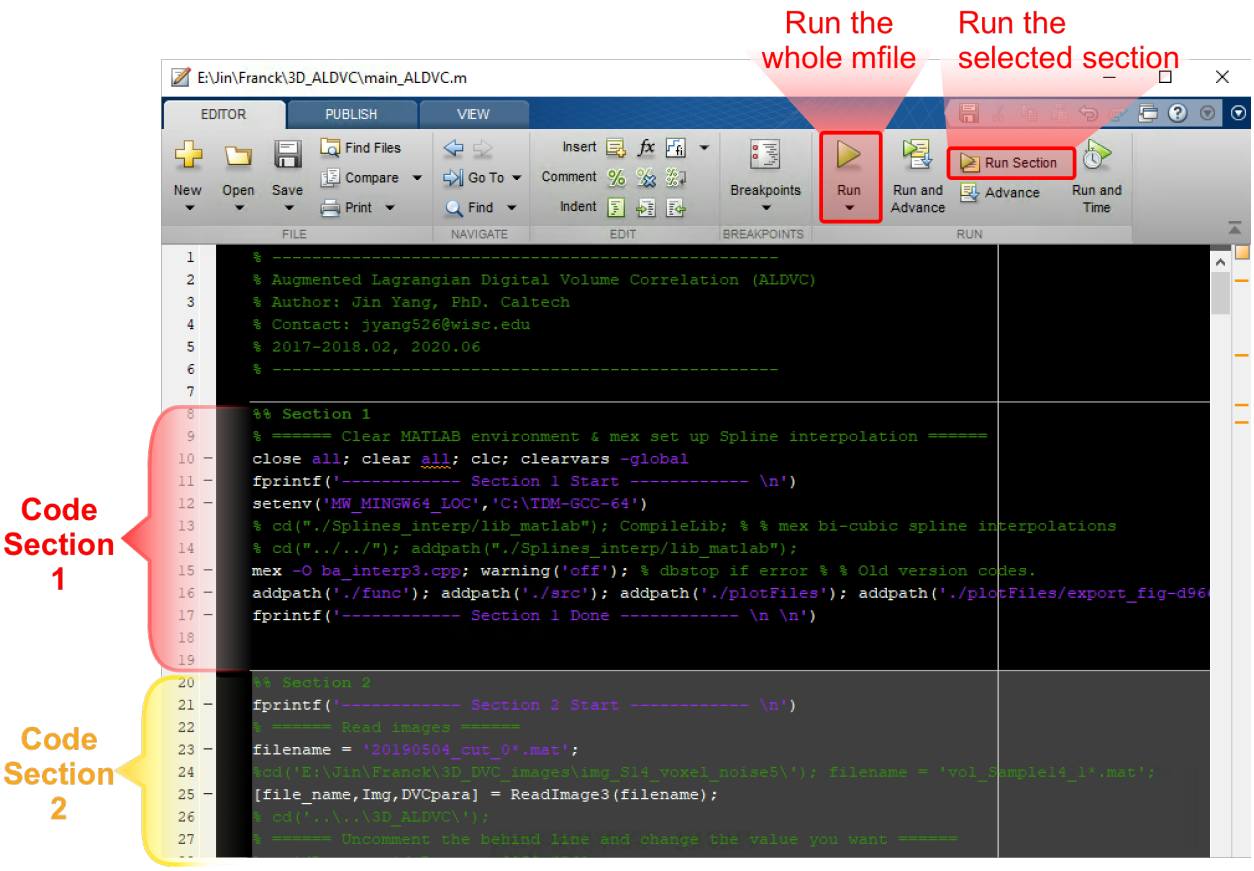


Figure 2: Main file of ALDVC code: “main_ALDVC.m”. Each section can be executed in order by clicking “Run Section”.

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_ALDVC.m”, as shown in Fig. 2, ALDVC code can be executed section-by-section. Once you are familiar with ALDVC code, you can execute the whole main file by clicking the ”Run” button to run the whole file at one time (”EDITOR >> RUN >> ▶ ”). ALDIC code is easy to modify based on user’s custom parameter choice. In this code manual, we will introduce ALDVC code section by section.

3 3D volumetric image stacks pre-processing

To apply digital volume correlation, we need to provide at least two image stacks to compare and register unknown deformations. These image stacks could be generated from various diagnostic techniques, e.g., 3D confocal microscopy, X-ray tomography (CT) scans, magnetic resonance imaging (MRI), neutron tomography, etc.

Here we assume the user has 3D image stacks for both undeformed and deformed images. (Though your raw data sets are not in image stacks, there are lots of free software, e.g., Fiji (https://imagej.net/Fiji), ImageJ or other image processing softwares, to export 3D image stacks.) For example, in the subfolder "./DVC_images/vol_stretch_tiff" there are one example of such a 3D image stack corresponding to the "vol_stretch_1001.mat" in the "ImageDownload_link.txt".

We provide a MATLAB script "./DVC_images/GenerateVolMatfile.m" to transfer such image stacks to ALDVC input matfiles. (User needs to modify this script a little bit where are marked with " `TODO:` " based on his/her data sets).

Currently, we recommend the user to save each 3D image stack into one "vol" MATLAB matfile, and always export the reference image as the first vol matfile.

Comments The "x-, y-, z-" or "1-, 2-, 3-" or "first, second, third" coordinates in the ALDVC code always correspond to the 1st, 2nd and 3rd indices of Matlab workspace variable. For example, `p_meas(:,1)` and `p_meas(:,2)` are the x- and y-coordinates of scattered points `p_meas` .

This is a little different from some MATLAB image processing functions. For example, if a 3D image has size M×N×L, in this code, we always have the image size with size_x = M, size_y = N, and size_z = L. If you use some MATLAB computer vision or image post-processing functions, for example, 'imagesc3D', or 'imshow3D', or 'surf', it will reads as size_x = N, size_y = M, and size_z = L.

At the end of the "./DVC_images/GenerateVolMatfile.m" and in the subfolder "./PlotFiles/", you may find that there are two functions "imagesc3.m" and "imagesc3D.m". As mentioned before, " `imagesc3(vol1);` " is equivalent to " `imagesc3D(permute(double(vol1),[2,1,3]));` "

Please pay attention to this difference.

4 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, tri-cubic (by default) and tri-cubic splines interpolations are implemented in this code. For example, by default we use tri-cubic interpolations where the associated mex set up file is called "ba_interp3.cpp" [7].

4.1 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 4.2 and jump to Section 4.3.

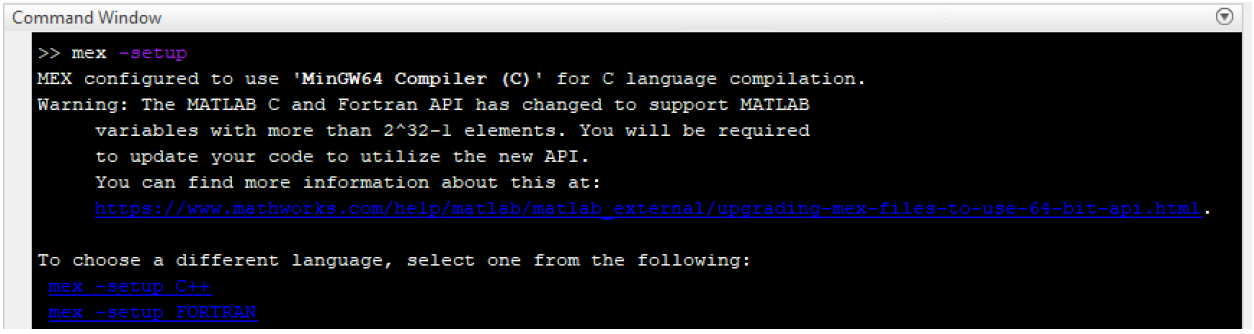


Figure 3: Message display on the command window when a mex C/C++ compiler is installed successfully.

4.2 Install mex C/C++ compiler

The step of installing the mex C/C++ compilers is a common step for users to run C/C++ codes with MATLAB. Mac users usually do not 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 [8, 9].

- Download: TDM-gcc compiler from: http://tdm-gcc.tdragon.net/
- Install TDM-gcc compiler on your computer. For example, I install it at 'C:\TDM-GCC-64' [2].
- 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.

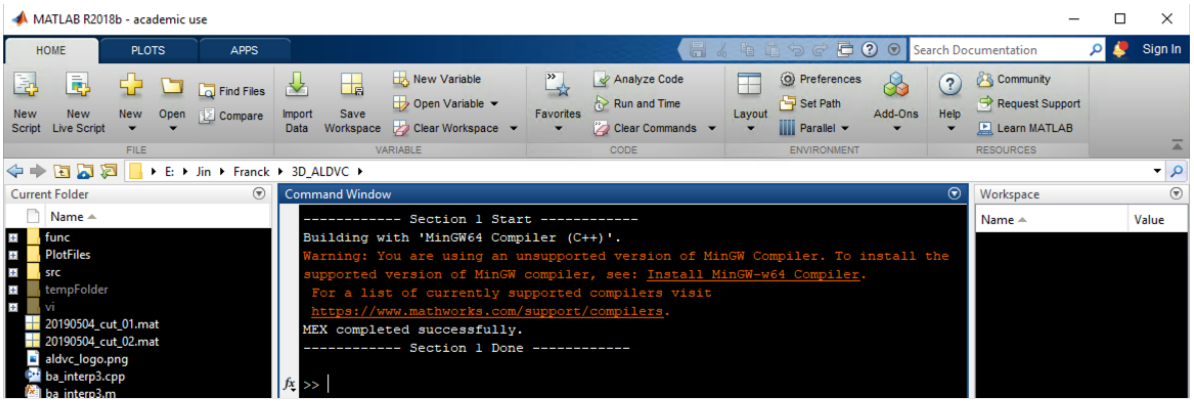
Alternatively, the MinGW mex compiler can be downloaded from the MATLAB 'Add-Ons' library and installed per the MATLAB documentation. In this case, comment out the setenv line in section 1.

4.3 Execute code Section 1

Once a mex C/C++ compiler is installed, we can execute main_ALDVC.m code Section 1 and a successful message will display on the MATLAB command window, see Fig. 4.

[2] 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.

(a)



(b)

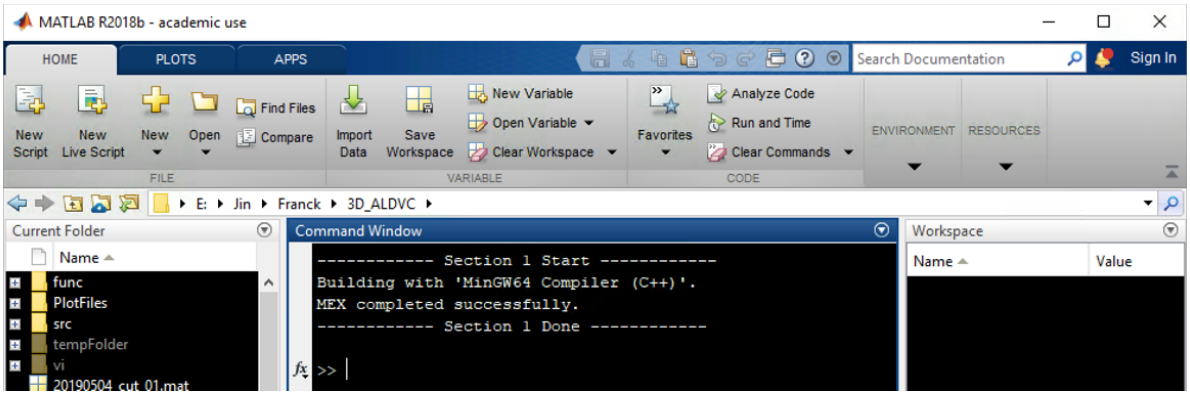


Figure 4: ALDVC code Section 1 is executed successfully and a message similar to (a) or (b) will display on the command window.

5 Code Section 2: Load images and set up DVC parameters & mesh

This section is to load DVC reference and deformed images and set up DVC parameters. Before running ALDVC, we need to transform all the volumetric images into MATLAB .mat files.

User is expected to transform volumetric images into MATLAB .mat-files. Each volumetric image is stored as a cell in each matfile (see example matfile "vol_stretch_1001.mat").

We recommend 3D volumetric DVC images have at minimum three times the subvolume size in each dimension. For example, if you want each local subvolume size to be 20 vx by 20 vx by 20 vx, the minimum input image size should be at least 60 vx by 60 vx by 60 vx [3]. Finally, do not forget to put your images on the MATLAB working path.

After executing this section, all the ALDVC associated parameters will be stored in the workspace in "DVCpara". All the mesh properties will be stored in "DVCmesh" after executing code Section 3. Here we show a brief summary of both these two data structures for an example case in Table 1 and Table 2.

5.1 Load DVC images

First, define your `filename` and `fileFolder` in code Section 2. For example, a pair of volumetric images (`'vol_stretch_1001.mat'` , `'vol_stretch_1003.mat'`) from a 10% uniaxial stretch test are stored in the subfolder `'./DVC_images'` . To load these images, we could define `filename = 'vol_stretch*.mat';` `fileFolder = './DVC_images/'` in code section 2. If your images are stored in the main path of the ALDVC code instead of in a subfolder, you can define fileFolder as an empty string (`fileFolder = ''`) or simply comment it (`%fileFolder = ...`).

5.2 Define region of interest (ROI)

When executing code in Section 2, the user can choose the whole volumetric image as the ROI, or define the region of interest (ROI) by directly clicking corner points on a popped out original image. On the command window, it displays

```
--- Open original images to click DVC domain? ---
Input here(0:Open; 1:Not Open):
```

If we input `1` , the whole 3D volumetric image will be used automatically. If we input `0` , an xy- and then an xz- slice of the volumetric image will pop out and user is expected to click both the left-top and right-bottom corner points on the first xy-slice and the second xz-slice to define image ROI (cf. Fig. 5). When clicking corner points, their coordinates will also display on the command window in the unit of voxels and stored in the variable `DVCpara.gridRange` .

```
--- Define top-left and bottom-right corner points on the xy-plane ---
xy-coordinates of top-left corner point are (37.000,37.000)
```

[3] If your image is not large enough to run ALDVC, our FIDVC code may be a suitable alternative [10, 11, 12, 13].

Table 1: Summary of DVC parameters in “DVCpara” structure

DVCpara variable	DVC parameter	Description and comments
winsize	Subvolume window size [winsize_x, winsize_y, winsize_z]	Local subvolume size in ALDVC Subproblem 1
winstepsize	Subvolume window step [winstepsize_x, winstepsize_y, winstepsize_z]	The distance between neighboring local subvolumes, or the finite element size in ALDVC Subproblem 2.
gridRange	DVC region of interest (ROI)	ROI can be defined by clicking corner points.
Subpb2-FDOrFEM	0-’FD’: Finite difference method; 1-’FEM’: finite element method	Both finite difference and finite element methods are implemented to solve ALDVC Subproblem 2.
ClusterNo	Number of threads to perform parallel computing	ALDVC Subproblem 1 can be sped up by applying parallel computing.
ImgSize	Image size	Size of 3D volumetric images
ImgSeqIncUnit	To decide to perform cumulative or incremental DVC mode	Cumulative DVC 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.
InitFFTMethod	Method to compute an initial guess: (0-’bigxcorr’) multiscale ZNCC; (1-’xcorr’; 2-’phasecorr’; 3-’bigxcorrUni’; 4-’bigphasecorrUni’) uniform ZNCC/phase correlation for small and large deformations, respectively	Methods 1 and 2 are good for measuring small deformations. Practically, if the magnitude of displacement is greater than 40% of the subset size, initial guess of such large deformation can be measured using methods 0,3 or 4.
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.
interpmethod	Interpolation scheme of volumetric image grayscale at subvoxels	Interpolation scheme can be chosen from {’linear’, ’cubic’(recommended), ’spline’}.
displayIterOrNot	Display convergence details of Subproblem 1 IC-GN iterations	0:Don’t display IC-GN iteration info; 1: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 0.01 vx by default.

Table 2: Summary of DVC mesh in “DVCmesh” structure

DVCmesh variable	DVC parameter	Description and comments
coordinatesFEM elementsFEM	Coordinates of nodal points in the finite element mesh and their connectivity	Linear 8-node hexahedron (HEX8) elements are used here. However, it can be extended to other type of finite elements with arbitrary shape functions.
dirichlet neumann	FE-mesh nodal points at the boundary	Indices of nodal points at ROI borders are assigned with Dirichlet or Neumann boundary conditions.
xyz0	Regular FE-mesh nodal grids	Only good for regular HEX8 FE-meshes.

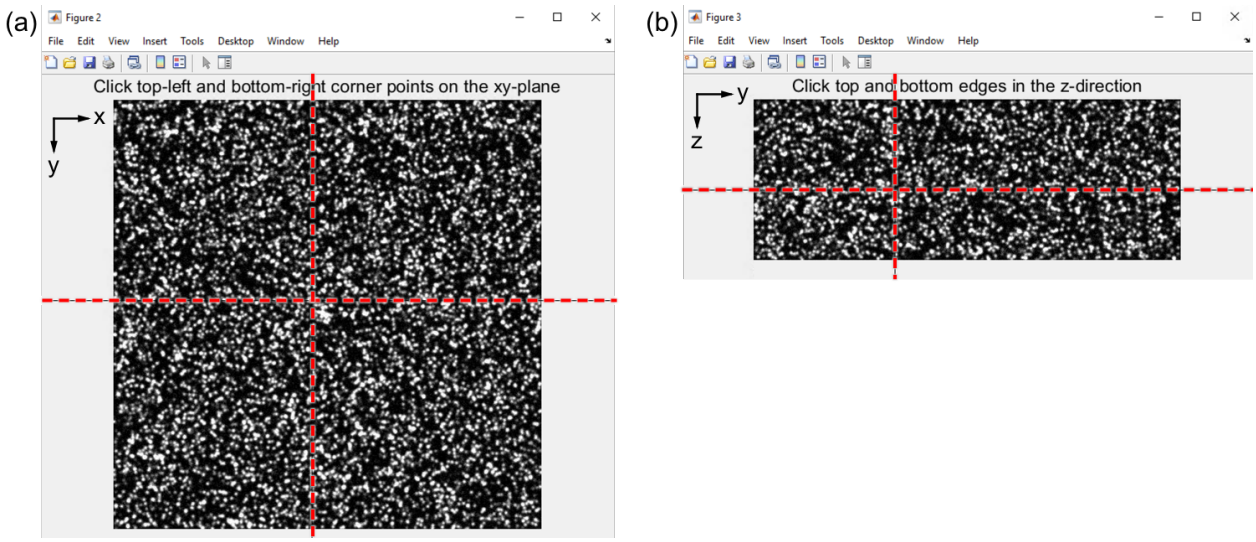


Figure 5: Click corner points on the (a) xy- and (b) xz-slices of DVC volumetric image to define coordinates of corner points of the region of interest(ROI).

```
xy-coordinates of bottom-right corner point are (481.000,480.000)
--- Define top and bottom edges in the z-direction ---
z-coordinate of the top edge is (63.000)
z-coordinate of the bottom edge is (66.000)
```

Comment: If a clicked corner point is out of the image border, ALDVC code will adjust it back to the nearest point along image borders. Thus, if all the corner points are clicked outside of the image, it will assign the whole image as the DVC ROI.

5.3 Set up DVC parameters

Then user will be asked to decide local subvolume size and the subvolume step. “Subvolume size” is the edge length of local subset window; while the “subvolume step” is the distance between two neighboring subvolumes, and choice of “subvolume step” can be independent of choice of subvolume size.

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

Subvolume size is the window size where we use to obtain initial guess through FFT cross correlation method, or through SSD minimization IC-GN iterations in the ALDVC local step. Subvolume step size is also the finite element mesh size in the ALDVC global step. Practically you can choose subvolume size between 10 vx by 10 vx by 10 vx and 30 vx by 30 vx by 30 vx. Try to nclude 3∼5 features of your volumetric images when choosing the size; subvolume step size can be selected as 0.25∼1 times of subvolume size, decreasing the step size increases the sampling frequency but also increases computational cost. For example, in the uniaxial stretch test case, we select subvolume size as `20` -or- `[20,20,20]` voxels, and subvolume step is chosen as `10` -or- `[10,10,10]` voxels [4].

We need to choose a method to compute an initial guess for the DVC problem. Here we provide five options and we will make a comparison of these five methods in Section 6. Without any prior information of the deformation field, choose the default method first.

```
--- Select initial guess method ---
    0: By default: no prior info of deformation field;
       Multiscale zero normalized cross correlation for large deformation;
    1: Zero normalized cross correlation for small deformation;
    2: Phase correlation for small deformation;
    3: Zero normalized cross correlation for large deformation (expensive
   !);
    4: Phase cross correlation for large deformation (expensive!);
Input here: 0
```

We also need to choose the solver method for ALDVC Subproblem 2 (global step). For this version of the code, we can only deal with uniform grid mesh, and both the “Finite difference method” and “Finite element method” solvers are nearly identical in practice. Even if you choose “Finite difference method”, there is still a finite element mesh generated (cf. “DVCmesh” in the Matlab workspace), which could help you conduct FEA analysis if you want to combine DIC with other FEA codes/software. My personal experience is that finite difference method is faster and a little bit more accurate than finite element method in the ALDVC Subproblem 2 (global step) because of the boundary effects.

```
--- Method to solve ALDVC 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 do not 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
```

[4] If the input subvolume size (`winsize`) or step (`winstepsize`) is a single scalar, ALDVC code will change it to an vector with three identical entries automatically.

If we do not want to use parallel pools, input `0` or `1` . If we want to use parallel computing, enter the number of parallel threads . E.g. Input `4` .

MATLAB parallel computing environment can be set in the "Home HOME ≫ Environment ≫ Parallel Parallel ≫ Parallel Preferences" (see Fig. 6 and [14] for more information about `parpool`).

5.4 Additional parameters setup when dealing with image sequence

If we use an image sequence with more than two frames, for each new frame we can choose to use the displacement results in last frame as the initial guess for the new image frame, or we can just redo FFT initial guess for every new frame. This choice depends on how big relative displacements can be between two consecutive frames and the monotonicity of the experiment. Generally, if the relative displacement field between two consecutive frames is smaller than 5-7 voxels, we can use the deformation result of last frame as the initial guess displacement field for the new frame; otherwise it is suggested that we still need to redo the FFT initial guess process.

```
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:
```

In most cases, cumulative mode works well unless the image pattern features have changed a lot. Here `cumulative` mode in this version of the code. Incremental mode will be implemented soon [5]. .

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

If you choose to use incremental mode (in future releases), you will further be inquired to input how often you would like to update the reference image:

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

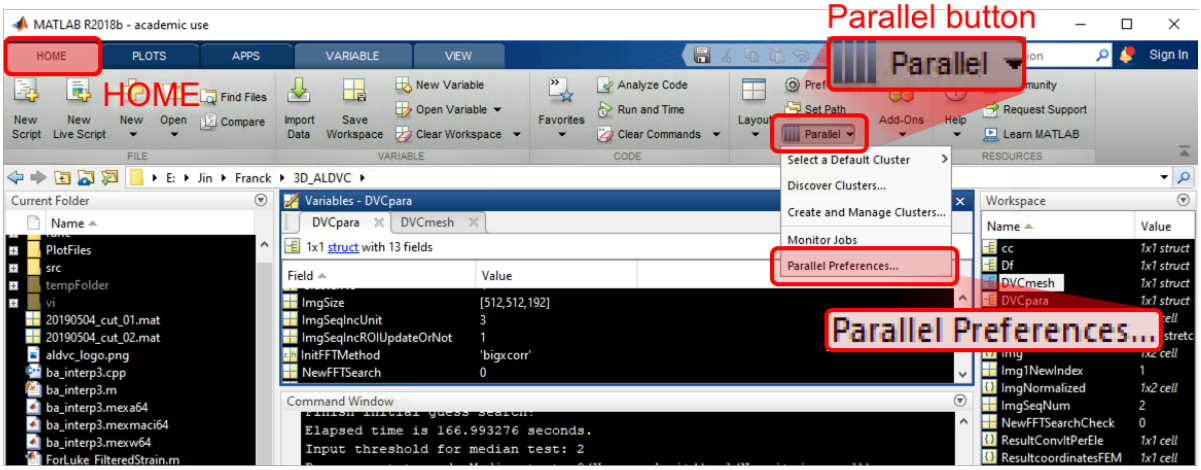
E.g., I want to update my reference image once every ten frames, so I input: `10` . The minimum number you can input is `1` , which means to update reference image every frame.

Every time the reference image is updated, you can choose to update the region of interest (ROI) at the same time or not. 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);
```

[5] Email us if you want more information: aldicdvc@gmail.com -or- jyang526@wisc.edu

(a)



(b)

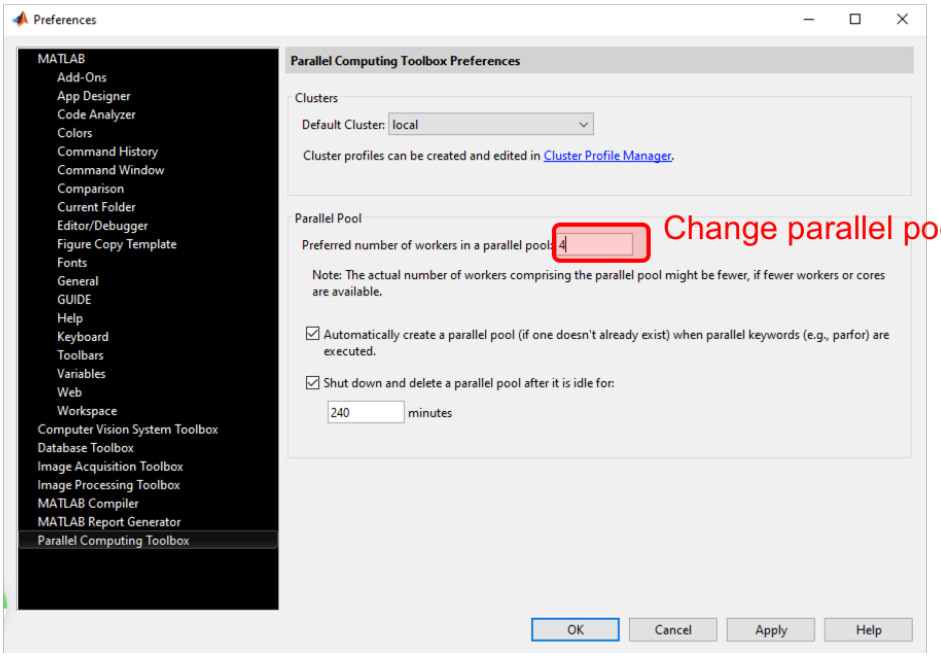


Figure 6: Set up MATLAB parallel preferences. See [14] for more information about `parpool` .

```
    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. Theoretically, this ROI update can be done automatically using the solved deformation of the last frame. However, we find it is most robust to update this ROI manually.

6 Code Section 3: Computing initial guess from FFT-based cross correlation

In this section, an initial guess of the unknown deformation is computed using fast Fourier transform (FFT) based method.

The user can choose one of the following methods to compute an initial guess. The command screen will display:

```
--- Select initial guess method ---
    0: By default: no prior info of deformation field;
       Multiscale zero normalized cross correlation for large deformation;
    1: Zero normalized cross correlation for small deformation;
    2: Phase correlation for small deformation;
    3: Zero normalized cross correlation for large deformation (expensive
   !);
    4: Phase cross correlation for large deformation (expensive!);
Input here: 0
```

6.1 FFT-based methods to compute initial guess

Here we provide five methods, see Fig. 7. For *small* deformations where displacement magnitudes are smaller than half subset size, please choose method 1 (zero normalized cross correlation, or ZNCC) or method 2 (phase correlation, or PC), see Fig. 8(c) and their associated correlation functions are shown in Eqs (1-2). In practice, method 1 is typically a little bit better than method 2.

$$\text{Zero normalized cross correlation function: } C_{\text{ZNCC}} = \frac{\int (f - \mu_f)(g - \mu_g)}{\sigma_f \sigma_g} \tag{1}$$

$$\text{Phase correlation function: } \quad C_{\text{PC}} = \frac{\mathcal{F}\{f\} \circ \overline{\mathcal{F}\{g\}}}{|\mathcal{F}\{f\} \circ \overline{\mathcal{F}\{g\}}|} \tag{2}$$

where $\mathcal{F}(\cdot)$ denotes 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. μ_f, μ_g are average grayscale values of DVC images f and g; σ_f, σ_g are the standard deviation of image grayscale values.

For *large* deformations where displacement magnitudes are larger than half the subset size, choose methods 0, 3-4. Methods 3-4 are the modified version of methods 1-2, where the comparing template for each local subset is in a larger area instead of comparing the same size subset

in the reference image, see Fig. 8(f). For large deformations, method 3 has the most accurate results, however it is usually very expensive. To speed up method 3, we build an image pyramid of the volumetric DVC images and solve method 3 in a multiscale way, which is method 0, see Fig. 8(g-h). The results of method 0 may be a little bit worse compared with method 3, however, it is much faster (cf. **Suggestions** in Fig. 7).

Comment: If there is large purely translational motion (motion with no rotation) but the overall distortion is much smaller, we would suggest the user to do pre-processing to remove the translational motion component before running ALDVC code (e.g. see image registration algorithms in MATLAB Image Processing toolbox or other rigid drift correction routines).

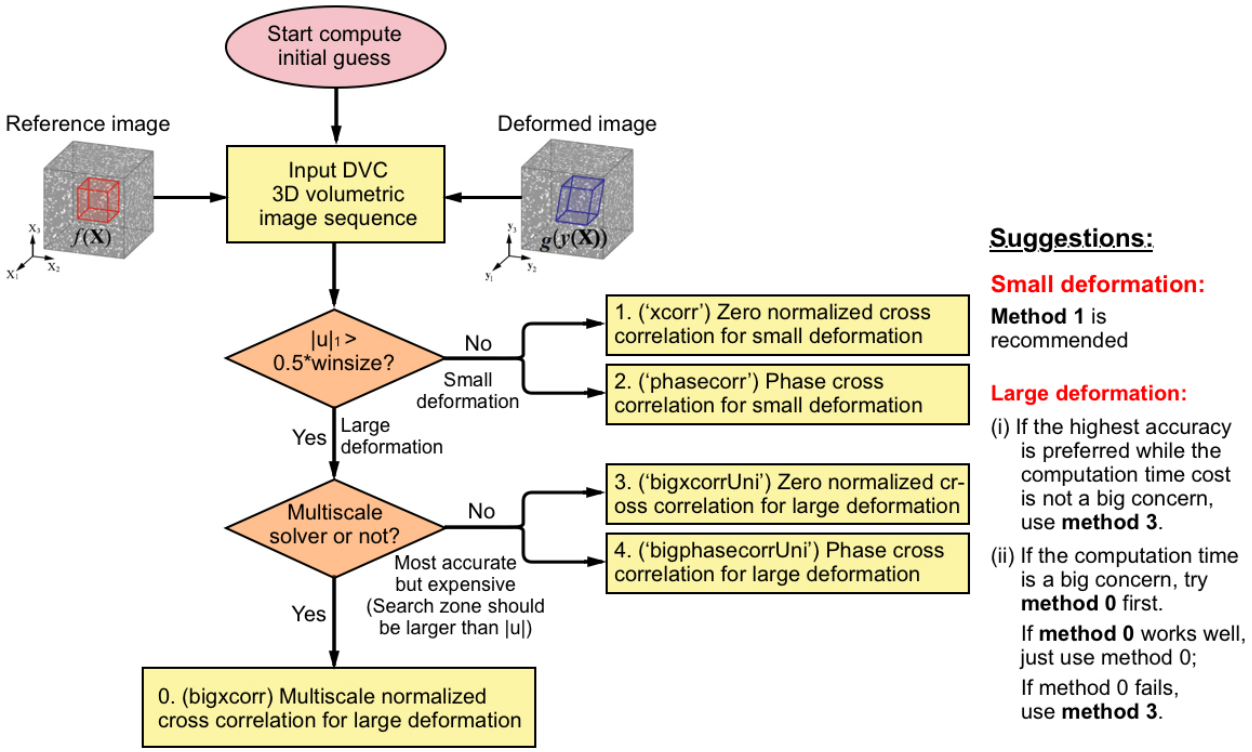


Figure 7: Flow chart for computing an initial guess of an unknown DVC deformation.

An example of computing the initial guess of a uniaxial stretch deformation is summarized in Fig. 8, where solved x-displacements on the xy-slice (with z=97 vx) are shown.

Methods 1 and 2 (cf. Fig. 8(a-b)) both work well where deformations are small (x$\in$[116, 346] vx). As a comparison, methods 0, 3-4 (cf. Fig. 8(d-e,g)) are able to solve the unknown deformation over the whole field, where method 3 has the best accuracy but lowest computation speed, see Table 3.

In summary, if deformations are small (u_x<0.5 winsize_x, u_y<0.5 winsize_y), method 1 is recommended. As for large deformations (u_x>0.5 winsize_x or u_y>0.5 winsize_y), if the highest accuracy is preferred and the computation time cost is not a big concern, method 3 is recommended. If the

computation time is a big concern, try method 0 first. If method 0 produces poor results choose method 3 instead.

Table 3: Time cost of computing initial guesses of a uniaxial stretch example using different methods. (Test PC: Processor Intel(R) Core(TM) i5-4670K CPU 3.40GHz, RAM 16.0 GB, OS 64-bit Windows 10. Volumetric image size: 512vx×512vx×192vx; each subset size: 20vx×20vx×20vx; neighboring subset distance: 10vx×10vx×10vx.)

Method name	Method 1 ('xcorr')	Method 2 ('phasecorr')	
Time cost (s)	88.4	92.1	
Method name	Method 3 ('bigxcorrUni')	Method 4 ('bigphasecorrUni')	Method 0 ('bigxcorr')
Time cost (s)	1738.2	562.6	169.1

6.2 Remove noise in computed initial guess

In practice, FFT-based method solved initial guesses[6] may have large noise. The user can remove these bad points by applying a median filter, setting a q-factor threshold and setting both upper and lower bounds of displacements. The user can also continue to remove bad points by directly clicking them on each image stack and then press " `Enter` " key. However, this step needs lots of manual work and is time consuming and annoying, especially for 3D volumetric data. This by-hand picking step is **not recommended**.

```
Input threshold for median test: 2
Do you want to redo Median test: 0(Yes, redo it!); 1(No, it is good!)
Input here: 1

% ======= Find bad initial guess points manually by setting bounds =======
Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
Input here: 0
What is your upper bound for x-displacement?100
What is your lower bound for x-displacement?-100
What is your upper bound for y-displacement?100
What is your lower bound for y-displacement?-100
What is your upper bound for z-displacement?100
What is your lower bound for z-displacement?-100
Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)1

% ======= Find bad initial guess points manually =======
Do you clear bad points by directly pointing bad points? (0-yes; 1-no)
Input here: 1
```

At the end of this section, a finite element mesh will be generated automatically.

[6] Some designed filters or iterative deformation method (IDM) can further improve the accuracy of initial guess, cf. [10, 11, 12, 13]. These are beyond the scope of this code manual. Additionally, accuracy of computed initial guess will further be improved after executing ALDVC ADMM iterations (code Sections 4-6).

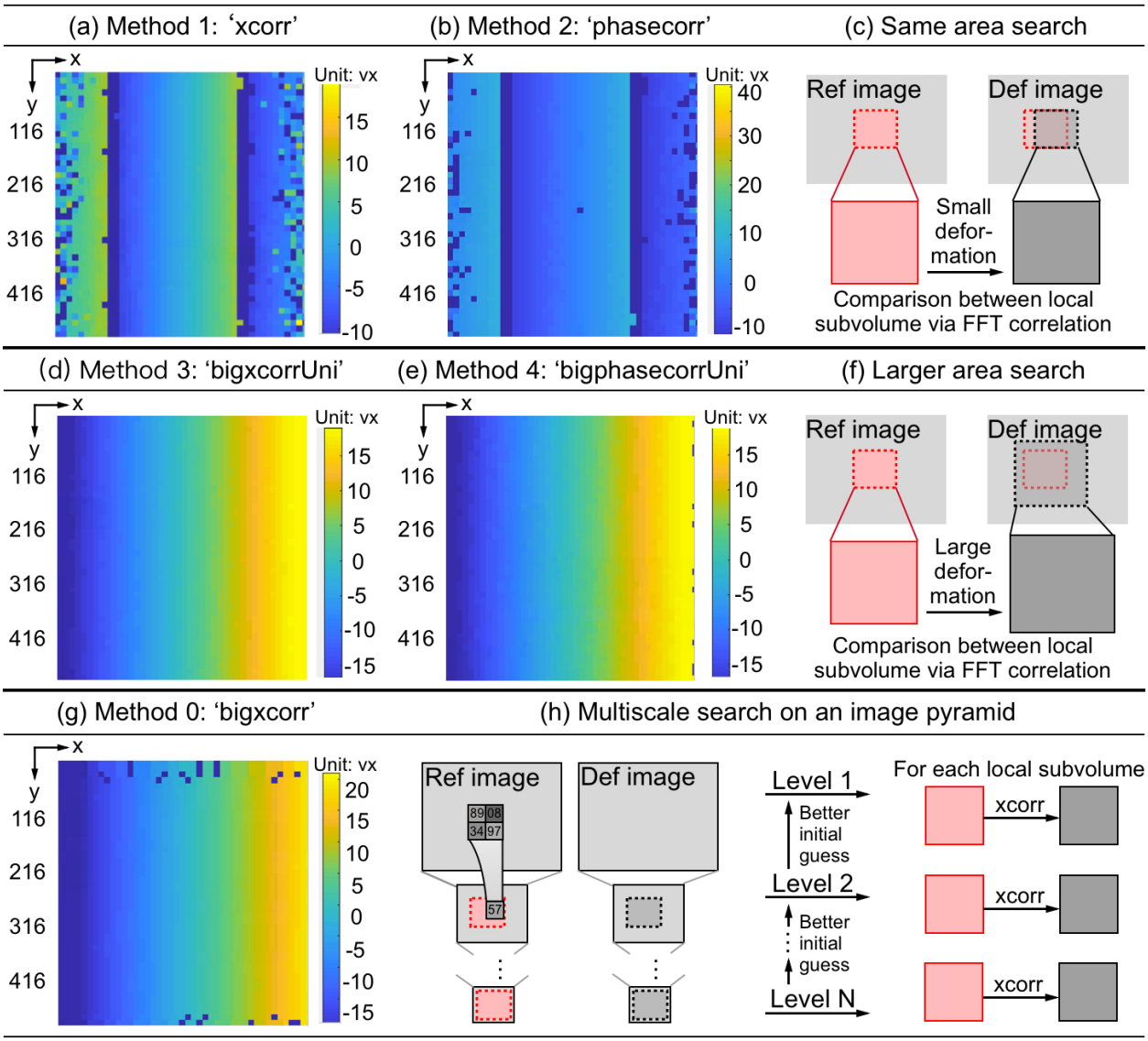


Figure 8: Example of computing an initial guess (x-displacements on the xy-slice with z = 97 vx) for uniaxial stretch via different FFT-based cross correlation methods. (a) Solved initial guess from zero normalized cross correlation ('xcorr'); (b) Solved initial guess from phase correlation ('phasecorr'); (c) Both methods (a-b) are good for measuring small deformations using same-area comparisons. (d) Solved initial guess from modified version of method (a) to measure large deformations ('bigxcorrUni'); (e) Solved initial guess from modified version of method (b) to measure large deformations ('phasecorrUni'); (f) Both methods (d-e) are good for measuring large deformations by being compared with larger area template image. (g) Solved initial guess from multiscale zero normalized cross correlation method ('bigxcorr'); (h) In method (g), an image pyramid is built and 'xcorr' method is solved in a multiscale way.

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

Image voxel grayscale value gradients will also be computed quickly using the finite difference operator and convolution operations.

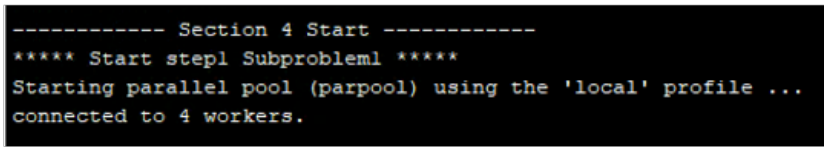
```
--- Computing image gradients done ---
```

7 Code Section 4: ALDVC Subproblem 1 (first local step)

7.1 Local subset IC-GN solver

In this section, we solve ALDVC ADMM Subproblem 1 (local step) using IC-GN (inverse compositional-Gauss Newton) scheme, where distributed parallel computing has been implemented. To execute this section, the IC-GN solver will work and a wait-bar will pop out automatically and allows user to visualize when a program will finish solving ALDVC Subproblem 1, see Fig. 9. If you have large DVC images, MATLAB parpool may spend several minutes before initializing the parallel computing and transferring the volumetric dataset. Don't worry and wait a little bit more if you don't see a waitbar immediately.

(a)



(b)

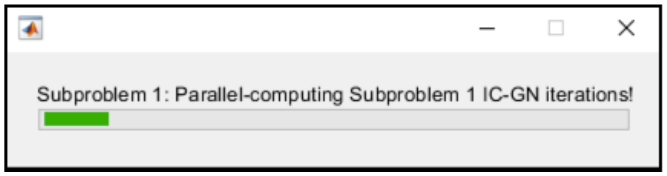


Figure 9: (a) Code section 4 starts and 4 threads are connected. (b) A waitbar pops out and allows user to visualize when a program will finish solving ALDVC Subproblem 1.

```
------------ Section 4 Start ------------
***** Start step1 Subproblem1 *****
Local ICGN bad subsets %: 585/26325=2.2222%
Elapsed time is 169.865714 seconds.
```

In this uniaxial stretch example, solved displacement and strain results are shown in Fig. 10 and Fig. 11. There are 585 (only 2.2% out of the total 26,325 subsets) bad local subsets that did not converge in the local step (IC-GN iterations) for this case. Here the PC is: Processor Intel(R) Core(TM) i5-4670K CPU 3.40GHz, RAM 16.0 GB, OS 64-bit Windows 10. Volumetric image size: 512vx$\times$512vx$\times$192vx; each subset size: 20vx$\times$20vx$\times$20vx; neighboring subset distance: 10vx$\times$ 10vx$\times$10vx. Four threads are used here. Although this computation speed is a little bit slower than the reported time cost in our paper [1] where an Intel i7-9800X, CPU 3.80 GHz, RAM 64.0 GB, 64-bit OS with 8 threads are used, it is still fast and finishing within three minutes with these DVC parameters. If you do not have the Parallel Computing toolbox installed in MATLAB, or the

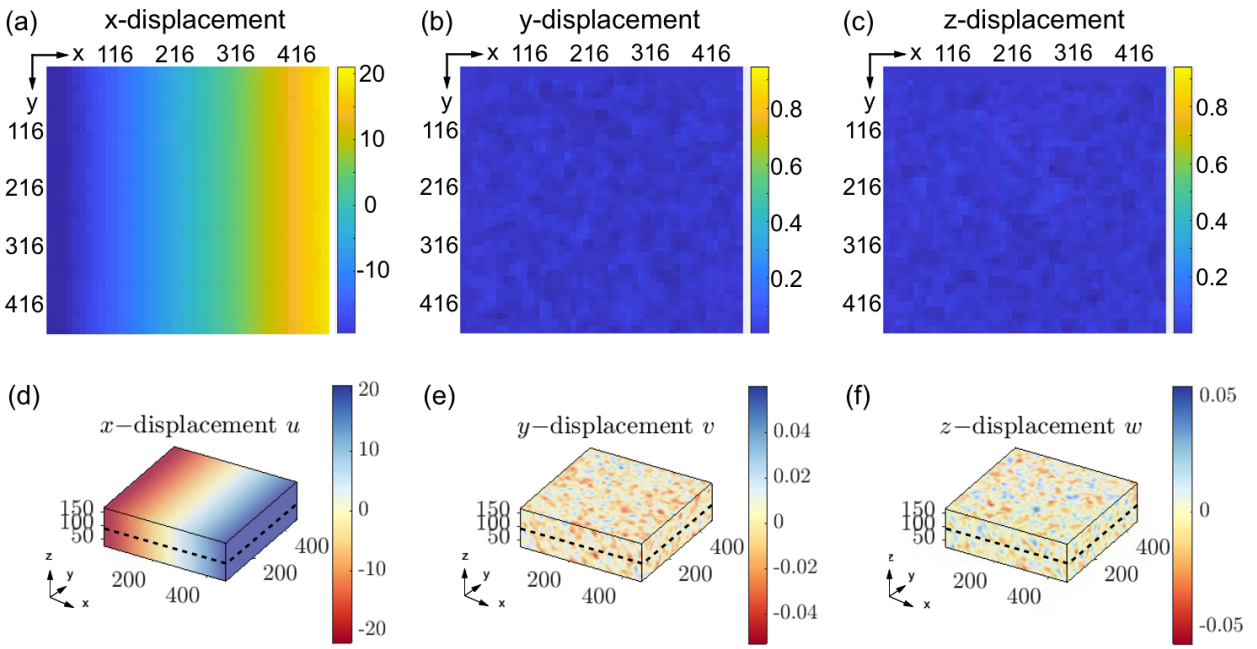


Figure 10: Solved displacements from Subproblem 1 IC-GN iterations. (a-c) Solved x-, y- and z-displacements on the xy-slice with z=97vx (black dashed lines in (d-f)). (d-f) 3D view of solved x-, y- and z-displacements.

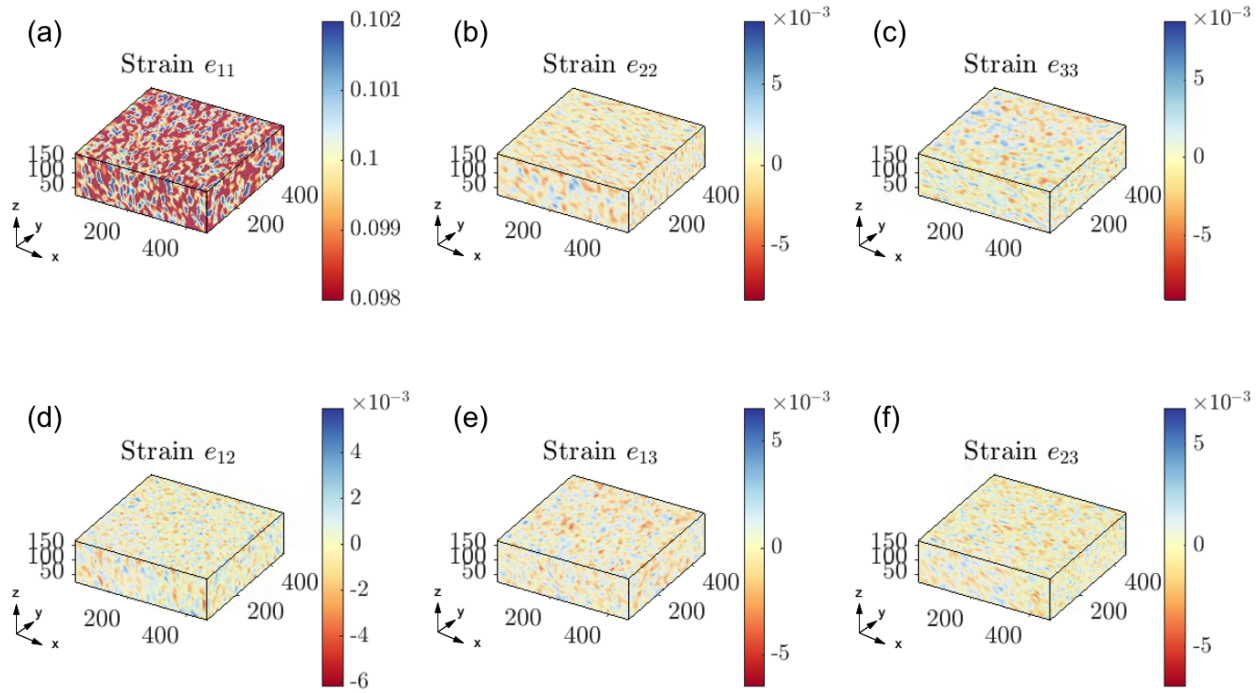


Figure 11: Solved infinitesimal strains from Subproblem 1 IC-GN iterations.

computer memory is limited, please modify the `DVCpara.ClusterNo` as `1` , which uses a single thread to compute ALDVC Subproblem 1.

7.2 Remove results of bad subsets

When this step finishes, a report will display on the command window. Same as Section 6, user can further remove these bad points by applying a median filter, setting a q-factor threshold and setting both upper and lower bounds of displacements. User can also continue to remove bad points by directly clicking them on each image stack and then press " `Enter` " key [7].

```
--- Start to remove bad local points ---
Input threshold for median test: 2
Do you want to redo Median test: 0(Yes, redo it!); 1(No, it is good!)
Input here: 1
Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
Input here: 1
Do you clear bad points by directly pointing bad points? (0-yes; 1-no)
Input here: 1
****** Finish removing outliers! ******
--- Remove bad points done ---
------------ Section 4 Done ------------
```

Comment: If you see that all the local subsets diverged, in another words the percentage of local ICGN bad subset is 100%, this usually happens because of the bad DVC parameter choice or bad DVC pattern quality. Here are a few steps that may help fix the poor convergence.

(i) **Check DVC parameters.** In DVC, each subset/subvolume is expected to have enough features to track their deformations. With larger subvolume size ("DVCpara.winsize"), the ALDVC ADMM Subproblem 1 will have higher convergence percentage. Typically, we hope there are at least 3$\sim$5 features (e.g., fluorescent beads) within each local subset/subvolume. Besides subvolume size, considering both computation speed and accuracy, the distance between neighboring subsets (DVCpara.winstepsize) is recommended to be 0.25$\sim$1 times of "DVCpara.winsize".

(ii) **Plot and check initial guess** computed from Section 3 makes sense using the following code: `Plotdisp_show3(USubpb2,DVCmesh.coordinatesFEM,DVCmesh.elementsFEM);` . For example, if the deformation is not small ($\max(u_x, u_y) > 0.5$winsize), try method 0 or 3 rather than method 1 or 2 in code Section 3. If method 3 is used to compute the initial guess of a large deformation, make sure the size of search zone ("DVCpara.SizeOfFFTSearchRegion") is larger than the magnitude of displacement vector ($\max(u_x, u_y)$).

(iii) **Change parameters of IC-GN solver.** For example, increasing `DVCpara.Subpb1ICGNMaxIterum` , or decreasing `DVCpara.ICGNtol` may help improve IC-GN iteration convergence.

(iv) Sometimes the poor IC-GN convergence happens because the input image is not large enough to run ALDVC. For example, if the subvolume size is 20 vx by 20 vx by 20 vx, the

[7] The user is allowed to directly click bad points on each image stack manually, however, this step needs lots of manual work and is annoying, especially for 3D volumetric data. This step is **not recommended**.

minimum input image size should be at least 60 vx by 60 vx by 60 vx. If that happens, our previous FIDVC code [10, 11, 12, 13] may perform better.

8 Code Section 5: ALDVC Subproblem 2 (first global step)

After solving the ALDIC ADMM Subproblem 1 local step, we run Subproblem 2 global step in this section. Both finite difference and finite element methods are implemented.

8.1 Finite difference method

To solve the Subproblem using the finite difference method, a sparse finite difference operator **D** is generated 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

$$\begin{aligned} \mathbf{F} &= \left\{ \cdots \quad F_{11}^{(i)} \quad F_{21}^{(i)} \quad F_{31}^{(i)} \quad F_{12}^{(i)} \quad F_{22}^{(i)} \quad F_{32}^{(i)} \quad F_{13}^{(i)} \quad F_{23}^{(i)} \quad F_{33}^{(i)} \cdots \right\}^{\mathrm{T}} \\ \mathbf{u} &= \left\{ \cdots \quad u_{1}^{(i)} \quad u_{2}^{(i)} \quad u_{3}^{(i)} \quad \cdots \right\}^{\mathrm{T}} \end{aligned} \tag{4}$$

```
------------ Section 5 Start ------------
***** Start step1 Subproblem2 *****
Assemble finite difference operator D
Elapsed time is 0.506468 seconds.
Finish assembling finite difference operator D
Elapsed time is 4.968799 seconds.
------------ Section 5 Done ------------
```

8.2 Finite element method

Subproblem 2 can also be solved using the finite element method. In this code, we use an 8-node hexahedron (HEX8) uniform finite element, the same element type with our global DVC code.

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

For uniform regular grid meshes, both the finite difference and the finite element method solve Subproblem 2 well and provide accurate results. In practice, the finite difference method behaves a little bit better than the finite element method in terms of computation speed and has better

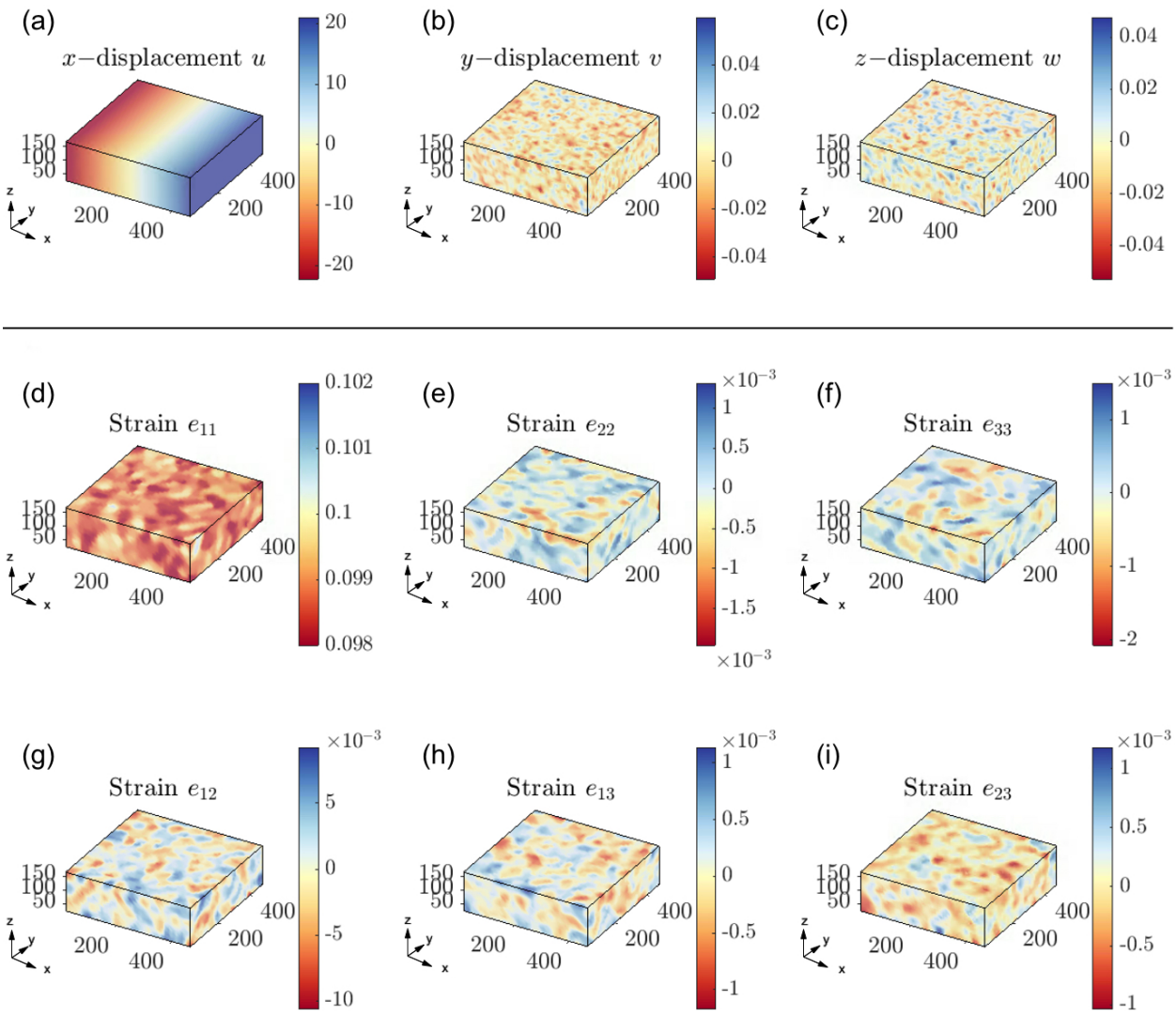


Figure 12: Solved displacements (a-c) and infinitesimal strains (d-i) from Subproblem 2 finite difference method.

accuracy near the ROI boundary. However, the finite element method is much easier to implement with an arbitrary mesh, and can be easily combined with an adaptive meshing technique [8].

9 Code Section 6: ALDVC ADMM iterations

In this section ALDVC iteratively uses the alternating direction method of multipliers (ADMM) iterations in this section.

[8] Arbitrary mesh has not yet been implemented yet for ALDVC. Welcome to email us or collaborate if you want more information.

The tolerance threshold of ADMM iteration is set to be `1e-4` by default. This threshold can be manually adjusted by modifying `tol2` in code Section 6. In practice, ALDVC ADMM usually converges within 3∼5 iterations. For example, in the uniaxial stretch example, ADMM iteration converges at the fourth step.

```
------------ Section 6 Start ------------
***** Start step2 Subproblem1 *****
Local ICGN bad subsets %: 585/26325=2.2222%
Elapsed time is 155.340947 seconds.
***** Start step2 Subproblem2 *****
Elapsed time is 0.665928 seconds.
Update local step  = 0.0032153
Update global step = 0.0031252
**********************************

***** Start step3 Subproblem1 *****
Local ICGN bad subsets %: 585/26325=2.2222%
Elapsed time is 152.020007 seconds.
***** Start step3 Subproblem2 *****
Elapsed time is 0.648582 seconds.
Update local step  = 0.00027555
Update global step = 0.00090898
**********************************

***** Start step4 Subproblem1 *****
Local ICGN bad subsets %: 585/26325=2.2222%
Elapsed time is 136.148046 seconds.
***** Start step4 Subproblem2 *****
Elapsed time is 0.668043 seconds.
Update local step  = 9.8178e-05
Update global step = 0.00035723
**********************************
```

10 Code Section 7: Check convergence

This section checks convergence of ALDVC ADMM iterations and deletes temporary variables in the workspace. This section can be skipped if you do not need it.

```
------------ Section 7 Start ------------
====== |u^-u| ======
0.0037007
0.0014816
0.00058201
0.00024628
====== |grad(u^)-F| ======
0.0019798
0.00018753
6.7804e-05
2.7535e-05
```

```
====== |delta u^| ======
0.0031252
0.00090898
0.00035723
====== |delta dual var udual| ======
0.00018753
6.7804e-05
2.7535e-05
====== |delta dual var vdual| ======
0.0014816
0.00058201
0.00024628
------------ Section 7 Done -------------
```

11 Code Section 8: Compute strains

11.1 Smooth displacement field if needed

Before computing strains, solved displacement fields can be further denoised if necessary. In most cases, ALDVC solved displacement fields are already denoised, so usually there is no need to further smooth solved displacement fields.

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

11.2 Compute strain field

In the ALDVC algorithm, "deformation gradient minus identity" **F** is a direct output in addition to the displacement field. Alternatively, the strain field can also be computed from numerically differentiating solved displacement field. We implement a total of four methods of computing the strain field.

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

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

(2) **Plane fitting method** to differentiate 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 ALDVC;
    1: Finite difference(Recommended);
    2: Plane fitting;
    3: Finite element;
Input here: 1
```

Three popular types of strains are implemented: infinitesimal strain, Eulerian strain based on the 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
```

11.3 Plot and save results

The user can decide to plot all the components individually or all together:

```
Plot displacement & strain components individually or all together?
    0: Plot each component individually;
    1: Plot all the deformation components together;
Input here: 1
Current image frame #: 2/2
------------ Section 8 Done ------------
```

Code Section 8 is easy to modify to save all the output figures as needed by the user.

```
% ------ Save figures ------
% Write down your own codes to save figures! E.g.: % print(['fig_dispu
    '],'-dpdf');
if strcmp(DVCpara.PlotComponentEachOrAll,'All')==1
    figure(1); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_disp.fig'
    ]);
    figure(2); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_strain.
    fig']);
elseif strcmp(DVCpara.PlotComponentEachOrAll,'Individual')==1
    figure(1); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_dispx.fig
    ']);
    figure(2); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_dispy.fig
    ']);
    figure(3); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_dispz.fig
    ']);
    figure(4); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_strainexx
    .fig']);
    figure(5); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_straineyy
    .fig']);
    figure(6); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_strainezz
    .fig']);
    figure(7); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_strainexy
    .fig']);
    figure(8); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_strainexz
    .fig']);
    figure(9); saveas(gcf,['fig_ImgSeqNum_',num2str(ImgSeqNum),'_straineyz
    .fig']);
else
```

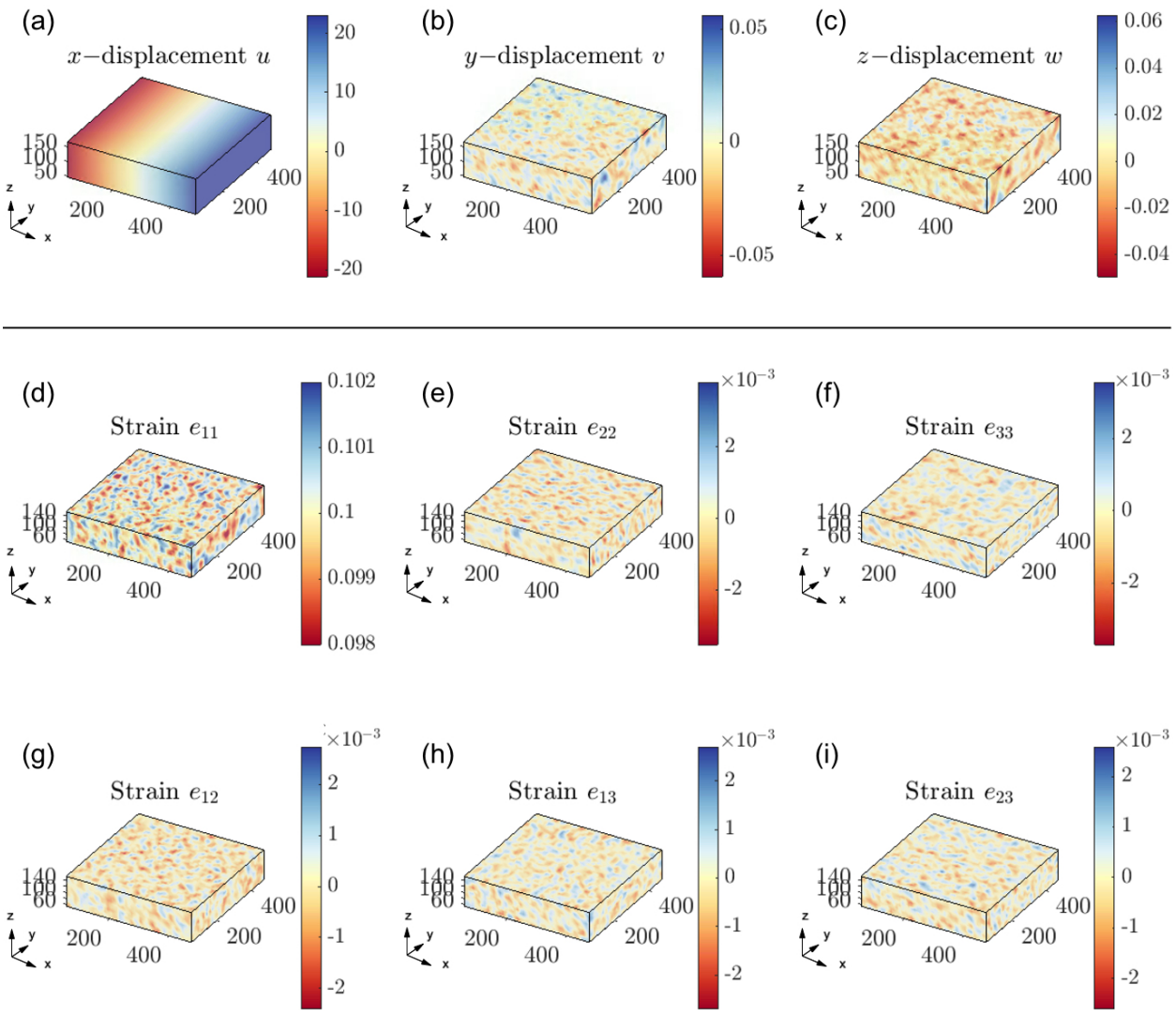


Figure 13: Final solved displacements (a-c) and infinitesimal strains (d-i) in the uniaxial stretch example.

```
    disp('=== Wrong input in DVCpara.PlotComponentEachOrAll! ===')
end
```

Finally, all the results will be saved in a Matlab matfile, see Table 4.

```
results_name = ['results_ws',num2str(DVCpara.winsize(1)),'_st',num2str(
    DVCpara.winstepsize(1)),'.mat'];
save(results_name, 'file_name','DVCpara','DVCmesh','ResultDisp','
    ResultDefGrad','ResultStrain','ResultFEMesh',...
     'ALSub1Time','ALSub2Time','ALSolveStep','Resultmubeta','
    ResultConvItPerEle', ...
     'ResultcoordinatesFEM','ResultelementsFEM','ResultSizeOfFFTSearchReg')
    ;
```

Table 4: Summary of DVC results

Variable	Description
DVCpara	ALDVC parameters, see Table 1
DVCmesh	ALDVC finite element mesh details, see Table 2
ResultDisp	Solved displacements from ALDVC
ResultDefGrad	Solved "deformation gradient minus identity" **F** from ALDVC
ResultConvItPerEle	Iteration steps of IC-GN scheme in solving Subproblem 1
ResultStrain	Solved strains from ALDVC code Section 8
Resultmubeta	Selected parameters μ,β in ALDVC
ResultSizeOfFFTSearchReg	Increased size of search area in computing initial guess
ResultFEMesh	Stored FE-mesh due to reference frame update in incremental mode
ALSub1Time, ALSub2Time	Computation time in solving Subproblems 1 & 2

12 Example 4.2 from ALDVC paper (SEM Challenge Sample 14)

Here, we compute another example modified from SEM Challenge Sample 14 (§4.2 from ALDVC paper [1]). These are characterized by a sinusoidal displacement field of varying frequency in the x-direction for three different frequency ranges (L1, L3, L5) at a fixed amplitude [15]. The synthesized 3D volumetric images of size 2048vx×192vx×192vx have the same x-direction deformations as the 2D images of the original challenge and have zero displacements in both the y- and z-directions. Here we set the subvolume size as 20vx×20vx×20vx and the window spacing step size to be 10vx×10vx×10vx.

Please modify the `filename` in ALDVC.m mfile as follows:

```
% ====== Read images ======
filename = 'vol_Sample*.mat'; fileFolder = './DVC_images/';
```

All the MATLAB command screen displayed messages are summarized here:

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

------------ Section 2 Start ------------
--- Open original images to define DVC domain? ---
Input here (0:Open; 1:Not Open): 0
--- Define top-left and bottom-right corner points on the xy-plane ---
xy-coordinates of top-left corner point are (11.500,13.500)
xy-coordinates of bottom-right corner point are (2035.500,181.500)
--- Define top and bottom edges in the z-direction ---
z-coordinate of the top edge is (10.000)
z-coordinate of the bottom edge is (180.000)

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

```
--- Select initial guess method ---
    0: By default: no prior info of deformation field;
       Multiscale zero normalized cross correlation for large deformation;
    1: Zero normalized cross correlation for small deformation;
    2: Phase correlation for small deformation;
    3: Zero normalized cross correlation for large deformation (expensive
   !);
    4: Phase cross correlation for large deformation (expensive!);
Input here: 1

--- Method to solve ALDVC 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

Since we are dealing with image sequences, for each new frame,
do we use last frame result as the initial guess or
Redo FFT initial guess for every new frame?
    0: Use last frame (by default);
    1: Redo initial guess.
Input here: 1

--- Choose cumulative or incremental mode ---
     0: Cumulative(By default);
     1: Incremental;
Input here: 0
------------ Section 2 Done ------------

Current image frame #: 2/4

------------ Section 3 Start ------------
Finish initial guess search!
Elapsed time is 151.077727 seconds.
Input threshold for median test (Default value: 2):
Do you want to redo Median test: 0(Yes, redo it!); 1(No, it is good!)
Input here: 1
Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
Input here:
Do you clear bad points by directly pointing bad points? (0-yes; 1-no(
   Recommended))
Input here:
****** Finish removing outliers! ******
--- Finish setting up mesh and assigning initial value! ---
------------ Section 3 Done ------------

------------ Section 4 Start ------------
```

```
***** Start step1 Subproblem1 *****
Local ICGN bad subsets %: 0/58667=0%
Elapsed time is 359.794726 seconds.
--- Start to remove bad local points ---
Input threshold for median test (Default value: 2): 2
Do you want to redo Median test: 0(Yes, redo it!); 1(No, it is good!)
Input here: 1
Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
Input here:
Do you clear bad points by directly pointing bad points? (0-yes; 1-no(
   Recommended))
Input here:
****** Finish removing outliers! ******
--- Remove bad points done ---
------------ Section 4 Done ------------

------------ Section 5 Start ------------
***** Start step1 Subproblem2 *****
Assemble finite difference operator D
Elapsed time is 1.147445 seconds.
Finish assembling finite difference operator D
Elapsed time is 11.194479 seconds.
------------ Section 5 Done ------------

------------ Section 6 Start ------------
***** Start step2 Subproblem1 *****
Local ICGN bad subsets %: 0/58667=0%
Elapsed time is 343.787632 seconds.
***** Start step2 Subproblem2 *****
Elapsed time is 1.473837 seconds.
Update local step  = 0.0027736
Update global step = 0.0028836
**********************************

***** Start step3 Subproblem1 *****
Local ICGN bad subsets %: 0/58667=0%
Elapsed time is 342.549464 seconds.
***** Start step3 Subproblem2 *****
Elapsed time is 1.409494 seconds.
Update local step  = 0.00025628
Update global step = 0.00086068
**********************************

***** Start step4 Subproblem1 *****
Local ICGN bad subsets %: 0/58667=0%
Elapsed time is 327.061197 seconds.
***** Start step4 Subproblem2 *****
Elapsed time is 1.355582 seconds.
Update local step  = 9.275e-05
Update global step = 0.00034032
**********************************
```

```
119
120 ------------ Section 6 Done ------------
121
122 Current image frame #: 3/4
123
124 ------------ Section 3 Start ------------
125 ------------ Section 3 Done ------------
126
127 ------------ Section 4 Start ------------
128 ***** Start step1 Subproblem1 *****
129 Local ICGN bad subsets %: 0/58667=0%
130 Elapsed time is 343.358540 seconds.
131 --- Start to remove bad local points ---
132 Input threshold for median test (Default value: 2): 2
133 Do you want to redo Median test: 0(Yes, redo it!); 1(No, it is good!)
134 Input here: 1
135 Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
136 Input here: 1
137 Do you clear bad points by directly pointing bad points? (0-yes; 1-no(
       Recommended))
138 Input here: 1
139 ****** Finish removing outliers! ******
140 --- Remove bad points done ---
141 ------------ Section 4 Done ------------
142
143 ------------ Section 5 Start ------------
144 ***** Start step1 Subproblem2 *****
145 Assemble finite difference operator D
146 Elapsed time is 1.100657 seconds.
147 Finish assembling finite difference operator D
148 Elapsed time is 11.007307 seconds.
149 ------------ Section 5 Done ------------
150
151 ------------ Section 6 Start ------------
152 ***** Start step2 Subproblem1 *****
153 Local ICGN bad subsets %: 0/58667=0%
154 Elapsed time is 312.054279 seconds.
155 ***** Start step2 Subproblem2 *****
156 Elapsed time is 1.374596 seconds.
157 Update local step  = 0.0028118
158 Update global step = 0.0029139
159 **********************************
160
161 ***** Start step3 Subproblem1 *****
162 Local ICGN bad subsets %: 0/58667=0%
163 Elapsed time is 339.267349 seconds.
164 ***** Start step3 Subproblem2 *****
165 Elapsed time is 1.346837 seconds.
166 Update local step  = 0.00025855
167 Update global step = 0.00086357
168 **********************************
```

```
169
170 ***** Start step4 Subproblem1 *****
171 Local ICGN bad subsets %: 0/58667=0%
172 Elapsed time is 292.162367 seconds.
173 ***** Start step4 Subproblem2 *****
174 Elapsed time is 1.360442 seconds.
175 Update local step  = 9.3353e-05
176 Update global step = 0.00034105
177 **********************************
178
179 ------------ Section 6 Done ------------
180
181 Current image frame #: 4/4
182
183 ------------ Section 3 Start ------------
184 ------------ Section 3 Done ------------
185
186 ------------ Section 4 Start ------------
187 ***** Start step1 Subproblem1 *****
188 Local ICGN bad subsets %: 0/58667=0%
189 Elapsed time is 340.103346 seconds.
190 --- Start to remove bad local points ---
191 Input threshold for median test (Default value: 2):
192 Do you want to redo Median test: 0(Yes, redo it!); 1(No, it is good!)
193 Input here:
194 Do you clear bad points by setting upper/lower bounds? (0-yes; 1-no)
195 Input here:
196 Do you clear bad points by directly pointing bad points? (0-yes; 1-no(
       Recommended))
197 Input here:
198 ****** Finish removing outliers! ******
199 --- Remove bad points done ---
200 ------------ Section 4 Done ------------
201
202 ------------ Section 5 Start ------------
203 ***** Start step1 Subproblem2 *****
204 Assemble finite difference operator D
205 Elapsed time is 1.089632 seconds.
206 Finish assembling finite difference operator D
207 Elapsed time is 11.142363 seconds.
208 ------------ Section 5 Done ------------
209
210 ------------ Section 6 Start ------------
211 ***** Start step2 Subproblem1 *****
212 Local ICGN bad subsets %: 0/58667=0%
213 Elapsed time is 319.723090 seconds.
214 ***** Start step2 Subproblem2 *****
215 Elapsed time is 1.326119 seconds.
216 Update local step  = 0.002798
217 Update global step = 0.0028944
218 **********************************
```

```
***** Start step3 Subproblem1 *****
Local ICGN bad subsets %: 0/58667=0%
Elapsed time is 317.856610 seconds.
***** Start step3 Subproblem2 *****
Elapsed time is 1.342351 seconds.
Update local step  = 0.00025571
Update global step = 0.00085431
**********************************

***** Start step4 Subproblem1 *****
Local ICGN bad subsets %: 0/58667=0%
Elapsed time is 295.102789 seconds.
***** Start step4 Subproblem2 *****
Elapsed time is 1.376836 seconds.
Update local step  = 9.2468e-05
Update global step = 0.00033788
**********************************

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

------------ Section 8 Start ------------
Do you want to smooth displacement? (0-yes; 1-no)1
What method to use to compute strain?
    0: Direct output from ALDVC;
    1: Finite difference(Recommended);
    2: Plane fitting;
    3: Finite element;
Input here: 1
Infinitesimal stran or finite strain?
    0: Infinitesimal stran;
    1: Eulerian strain;
    2: Green-Lagrangian strain;
    3: Others: code by yourself;
Input here: 0
Plot displacement & strain components individually or all together?
    0: Plot each component individually;
    1: Plot all the deformation components together;
Input here: 1
Current image frame #: 2/4
Current image frame #: 3/4
Current image frame #: 4/4
------------ Section 8 Done ------------
```

Solved displacement and strain components are summarized in Fig. 14 and Fig. 15. Saved results and figures are stored in folder “ `./DVC_results/Sample14_noise5/` ”.

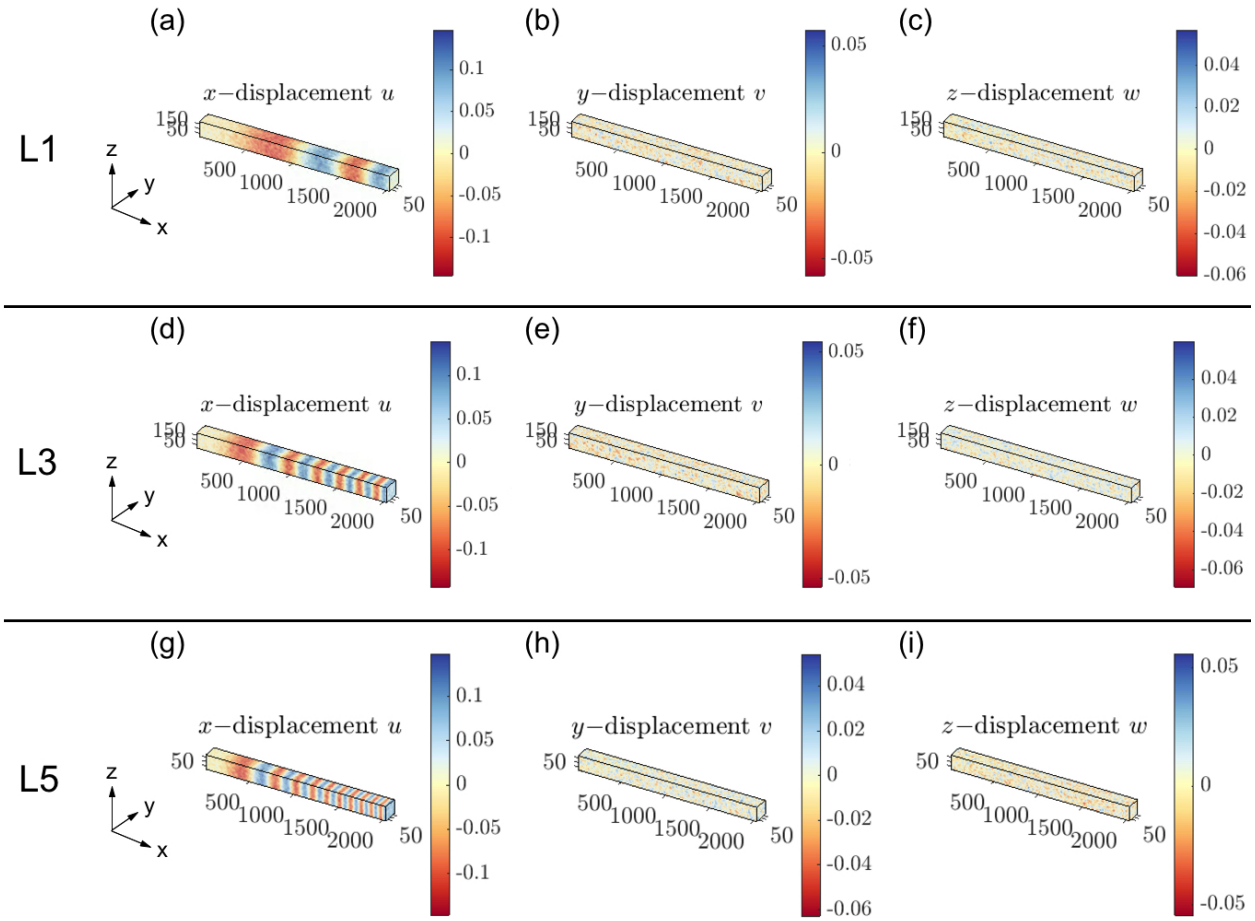


Figure 14: Final solved displacements of Sample 14 examples: (a-c) L1 (d-f) L3, (g-i) L5.

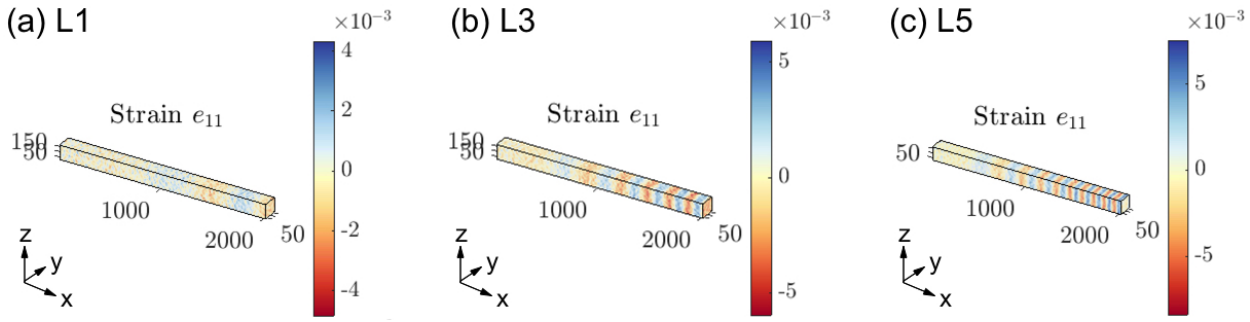


Figure 15: Final solved infinitesimal strain e_{11} of Sample 14 examples: (a) L1 (b) L3, (c) L5.

Acknowledgements

We gratefully acknowledge funding support from the Office of Naval Research (Dr. Timothy Bentley; grant N000141712058) and the National Institutes of Health (grant R01 AI116629). The au-

thors thank Prof. Kaushik Bhattacharya, Dr.Mohak Patel, and Dr. Orion Kafka for helpful discussions.

References

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

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

[3] *2D ALDIC code*. https://www.mathworks.com/matlabcentral/fileexchange/70499-augmented-lagrangian-digital-image-correlation-and-tracking.

[4] 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.

[5] https://www.researchgate.net/publication/342182706_Augmented_Lagrangian_Digital_Volume_Correlation.

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

[7] *3D Volume Interpolation with ba_interp3*. https://www.mathworks.com/matlabcentral/fileexchange/21702-3d-volume-interpolation-with-ba_interp3-fast-interp3-replacement.

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

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

[10] 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.

[11] 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.

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

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

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

[15] PL Reu, E Toussaint, E Jones, HA Bruck, M Iadicola, R Balcaen, DZ Turner, T Siebert, P Lava, and M Simonsen. DIC challenge: Developing images and guidelines for evaluating accuracy and resolution of 2D analyses. *Experimental Mechanics*, 58:1067–1099, 2018.