ClusterAlign User Manual – version 2022may09

(Program written by Dr. Shahar Seifer, Elbaum lab, Weizmann Institute of Science)

1. Purpose of the software

Transmission electron microscopes are often utilized for tomography based on a series of projection images (tilt series), followed by alignment and 3D reconstruction. Numerous software tools are available for these post-acquisition processing steps, including IMOD, bsoft, EMAN2, TomoJ, and others. The mechanism of rotation always introduces some jitter that must be corrected by alignment of the images before (or during) 3D reconstruction. Additionally, electron-optical distortions as well as geometric distortions due to beam exposure may need to be corrected for a successful reconstruction. In order to improve the alignment, nanoparticles are often distributed over the sample to serve as fiducial markers. Despite significant advances in software, precise tracking of fiducials can still be a difficult process to automate. This is especially so in thick samples suitable for STEM tomography, where individual nanoparticles may be shadowed or hidden by the specimen itself and therefore their visibility may differ drastically from one image to the next. ClusterAlign has been developed based on tests with such difficult datasets. The principle of operation is based on analysis of coherent clusters of fiducial markers rather than individual particles. Moreover, the structure of fiducial clusters is compared between the zero-tilt projection and all other projections without any assumption on correlation between adjacent tilts. Alignment error is mainly translational and thus expected to be corrected by solving for $\Delta x,\Delta y$ shifts of the image. Given a sufficiently large number of fiducial markers there will be points that share a similar height in z; in this case their relative locations in x,y are predictable according to a rigid body rotation and it is sufficient to track some of the markers uniquely in order to use them as anchors. Note that clusters are defined by a common height with respect to the tilt axis and need not be composed of apparently adjacent fiducials in any given projection; however, the vicinities in z and in x,y are often correlated so a cluster size limit may be imposed. The aim of the software is to optimize translational alignments under conditions of problematic fiducial visibility without manual intervention. Further optimizations such as focused local alignment or global distortion corrections or fixing additional degrees of freedom may be performed in other dedicated software such as tomoalign, EMClarity, or IMOD.

The input of the ClusterAlign software is a tilt series in mrc file format (or tif format) and a text file containing the rotation angles. The output includes the realigned tilt series in mrc format and text files with the fiducials locations and filled-in (“no-gaps”) locations, according to the standard format used in IMOD software. Gaps are filled by fitting to a model of a fixed rotation axis with minimum error, and the fitting errors are reported to a file. Optionally the 3D reconstruction is generated as well.

2. Installation

The installation in Windows is straightforward.

Windows

In windows, download the zip file found in:

https://sourceforge.net/projects/clusteralign/files/.

Extract the folder to your location of choice and run the setup.exe file. Then continue to install the Matlab scripts (see Optional Matlab toolboxes below).

The application makes use of

Nuget packages of EMGU.CV (https://www.emgu.com/wiki/index.php/Main_Page) wrapper of OpenCV (https://opencv.org/).

Avalonia (https://avaloniaui.net/) for user interface.

DotNet cross-platform framework by Microsoft (https://dotnet.microsoft.com/en-us/).

Details on the GPL3 license are here.

Linux / Mac

Download the source code (written in C#) based on a Visual Studio 2019 project found in https://github.com/Pr4Et/ClusterAlign.

Install the .NET 6 SDK framework on your platform, which enables building and running our source code without using Visual Studio:

https://dotnet.microsoft.com/en-us/download/dotnet/6.0

ClusterAlign depends on EMGU, which unfortunately does not provide tailored binaries for unmanaged code adapted from OpenCV and many contributors. So the entire source code from EMGU must be downloaded using GIT, compiled on your platform using CMAKE, and connected to ClusterAlign project. When downloading the EMGU source code using GIT make sure that you are located at a root folder that contains the ClusterAlign folder. Instructions are found in the chapters on GIT and Linux/Mac installation here:

https://www.emgu.com/wiki/index.php/Download_And_Installation

Note that building the EMGU.CV code takes a long time and should be monitored for faults. We have found that QT4 (emphasis on version 4) must be installed before running the CMAKE script. On Ubuntu the issue is solved using instructions in https://ubuntuhandbook.org/index.php/2020/07/install-qt4-ubuntu-20-04/.

For building/running the project, remove the multiple project files (*.sln, *.csproj) and leave only Linux.csproj. Enter the ClusterAlign_source folder and type

dotnet run

If there are still errors related to missing library files, enter on the command line: ldd * in folders with the object files from OpenCV to find which libraries are missing.

Optional Matlab toolboxes

The optional 3D reconstruction code, which generates 3D file from the aligned file, requires Matlab and a script called clusteralign_astra_reconstruct.m that we supply with the installation. In addition, the following external libraries should be downloaded:

Astra toolbox –Matlab version (https://www.astra-toolbox.com/),

and Matomo (https://bio3d.colorado.edu/imod/matlab.html).

The paths where the Matlab script and the libraries are located should be registered in the search list of Matlab by clicking Home-> Set Path -> Add with Subfolders.

For Linux, further installation instructions supplied with the Astra toolbox are needed to be followed precisely for the astra-toolbox to work. In Windows, no further steps are needed.

3. Preparation

The tilt series should be examined to determine the rotation axis (nearby x or y), the intensity values of fiducials relative to the sample (brighter or darker, to accommodate for dark field or bright field imaging, respectively), the typical size of fiducials in pixels, and their approximate number. If the rotation axis is expected to lay not close to x or y axes then the stack should be rotated in the x-y plane, in advance, (for example, using ImageJ->Image->Transform->Rotate). ClusterAlign is tolerant to deviations of up to 15 deg in the rotation axis.

Compatibility with IMOD is strongly desirable and recommended for tailored processing options and eventually building an optimal 3d reconstruction. IMOD corrects for the inverted y axis used in Thermo Fisher tomography software. Therefore, to avoid confusion it is suggested to convert the tilt-series file to IMOD recognizable using 'Create fixed stack' and 'Use fixed stack' in ETOMO before starting the ClusterAlign. The aligned file is later recognizable by ETOMO/IMOD based on the IMOD stamp placed in the file header. Also the tilt-angle file *.rawtlt is generated automatically by ETOMO, but it is a simple text file to edit.

If fiducials are very sparse, meaning that fewer than 5 fiducials can be detected and clustered in certain projection images, the fallback plan is to have the tilt series coarse-aligned in IMOD first. The file can be prepared with proper contrast after editing the prenewst.com file found in the IMOD working folder, then changing ModeToOutput from 0 to 2 (16 bits instead of 8 bits), and generating the file filename_preali.mrc using:

submfg prenewst.com

Note that ClusterAlign only refines the alignment of slices in which the fiducials can be detected, clustered and tracked.

4. Operation

The settings form (as shown in Fig.1) is opened once when the program starts and should be filled as follows.

Clicking the upper button <…> opens a file explorer to choose the tilt series in mrc or multi-tif format. Accordingly, the name of the path and the tilt-angles file are automatically determined; however, you should check the names and correct manually if needed. The tilt-angles file is a text file with angles in degrees in separate lines according to the order of the images, e.g., the .rawtlt file produced by standard software.

In the settings form choose Cluster radius in which, on average, several dozens of fiducials can be found. Usually a fifth of the image size is sufficient (400 pixels if your image is 2048X2048), but the value should be increased to the full image size if the fiducial spread is sparse (1000-2000 pixels).

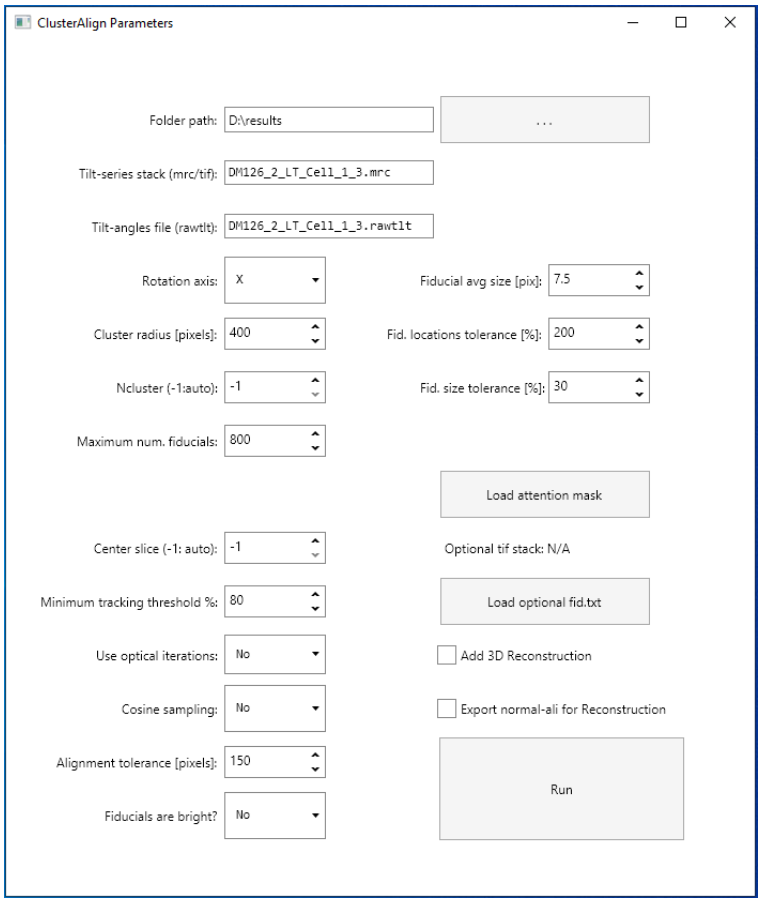

Figure 1. ClusterAlign Settings form

Enter (-1) in the center slice field to determine automatically the center slice number according to minimum tilt angle. This slice is the crucial one since fiducials not detected in this image are ignored entirely, and this slice is treated as reference for the modeled structure of the clusters.

Choose Ncluster=-1 for automatic search of best number of fiducials to define a cluster. If you choose manually, the tracking will be more robust for a larger number, at the expense of computation time and lower number of tracked fiducials reported.

Enter the maximum number of fiducials in the image (preferably even twice that number). If the value is too low not all the markers can be detected. If the number is much higher than the actual count then many artifacts will be marked as fiducials and the computation time will increase.

Enter the average size of fiducials (or -1 for automatic detection, when optical iteration is active). The fiducial locations tolerance is typically between 200%-300% and specifies how precisely the structure of the cluster is preserved during rotation. The tolerance is required partly to account for different heights (locations along the z axis), and partly due to possible distortions in rigidity of the sample. The fiducial size tolerance determines the variations in size (the range below and above the average value), which is relevant only for the initial composition of the fiducial template.

The minimum tracking threshold refers to a fiducial acceptance filter based on the percentage of slices where a tracked fiducial must be detected. Entering 100% (not recommended) means that all fiducial locations are tracked without any misses, while a smaller number (60-80% recommended) permits absence of validated fiducial locations, which implies that the fiducial locations are not reported in all slices. If the program reports “attempt fails” consider reducing that number and increasing the Cluster radius.

Use optical iteration=yes for attempting to improve results (at the expense of about double the processing time) by placing detection bias on the trajectory of tracked fiducials found in the first iteration. In the second iteration the tracking threshold will be increased automatically to achieve tracking in more slices at the expense of lower number of tracked fiducials. It is recommended to first run the program with optical iteration=no, then if improvement is needed you can add iteration after the program ends by starting the program again, setting optical iteration=yes and loading the fid.txt file from the previous iteration.

Cosine sampling=no is the default choice when the images are described by pixels of square aspect ratio, for example those acquired by a camera. Use Cosine sampling=yes if the aspect ratio varies according to cos function of the tilt angle (such option is available in our SavvyScan system, https://doi.org/10.1017/S1431927621012861). Note that in the former case, the region of interest in a given slice is limited to the part projected from the center slice. Outside this region, which is determined by the axis of rotation and cos of the tilt angle, the fiducials are not marked nor tracked because it will not be possible to find them in the center slice.

The alignment tolerance constrains the alignment shifts to a maximum (in x and y) according to the specified value. If you choose a tilt-series stack already after pre-alignment in IMOD then the tolerance is usually smaller than 20 pixels, otherwise we use 150 pixels.

In rare cases certain features in the sample may be identified inappropriately as fiducials (for example, points emerging from Bragg diffraction that are not anchored in the sample). In this case use the button <Load attention mask> to load a TIF stack that is similar in size to the tilt-series stack, where the regions of interest are marked with color 0 (black). The optical detection of fiducials is then limited only to spots painted in black in the attention mask. Such TIF stack can be prepared in ImageJ by loading the stack and saving as .attention.TIF file, choosing black ink using Edit->Options->Colors, and using the paintbrush to mark the regions of interest in every slice.

The button <Load optional fid.txt> can be used to review a previously generated file by the software. In this case the program will skip the optical detection and tracking analysis and only circle the fiducials that were tracked previously. The rest of the analysis will be performed rapidly as before.

The output of the analysis includes generated files, graphical window, and console output. For every slice a template image is shown shortly (Fig.2), which is generated based on average of

potential fiducial markers. Then the slice image is displayed (Fig.3), where the locations matching the fiducial template are marked with red circles. In the 'second optical iteration' green circles appear around the locations of tracked fiducials, and the circle color changes to white if the location is expected by the fitted model while the fiducial identity is still not confirmed. The size of the circles represent the fit error with additional margin that serves as region of interest with enhanced sensitivity for seeking additional markers.

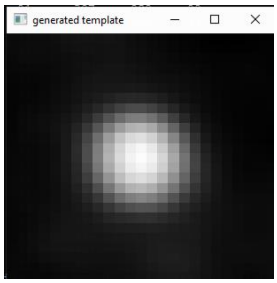

Figure2 . Template generated from average of suspected fiducial markers

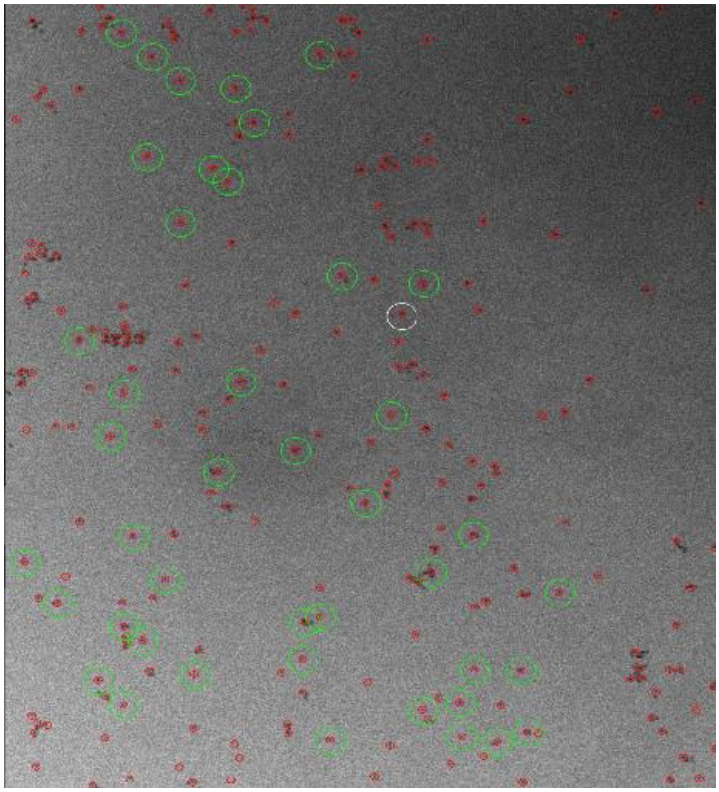

Figure3 . Detected fiducials are marked in red, tracked fiducials circled in green (cluster confirmed) or white (expected but yet unconfirmed).

Reports of the fiducial locations are shown in the console and in a file with extension .fid.txt, and have the following structure:

<object number> <fiducial number> <x location> <y location> <slice number>

The fiducial locations in x,y are provided per slice and per specific fiducial ID with respect to the original frames. Information on the z values, fit error, and other results appear in additional files. At the end of the analysis the aligned tilt-series is saved as a filename_jali.mrc (jali stands for 'just aligned'), and a summary including the rotation axis information is saved in filename.output.txt. Optionally, a standard filename_ali.mrc can be generated for

convenient reconstruction with external software by selecting *export normal-ali* in the settings. The aligned images are then rotated in the X-Y plane so the rotation axis of the sample appears vertical.

5. SIRT-3D reconstruction

Single axis

The Astra toolbox can be acquired and imported to Matlab as described in section 2.

Our Matlab function clusteralign_astra_reconstruct.m uses the SIRT3D algorithm in Astra toolbox to reconstruct a 3D stack smoothly from the jali.mrc file, the tilt-angles file, and the in-plane and out-of-plane angles of the rotation axis, and optionally with the fit-error file to exclude automatically tilts suffering a large error. The command to run in Matlab is written in the output.txt file generated at the end of the ClusterAlign analysis, so you can just copy and paste to Matlab. The process requires GPU card with sufficient memory (8Gb for 1024X1024X400 output stack). For example, the following command generates a rec_SIRT.mrc file in the case of horizontal rotation axis:

clusteralign_astra_reconstruct(0,90,0,'filename_jali.mrc','filename.rawtlt', 'filename.fit_err_by_slice')

Depending on your Matlab installation, the process can be activated automatically by marking the checkbox 'Add 3d reconstruction' in the user setting of ClusterAlign. However, by running the script manually it is possible to define a custom binning number, bin, and a custom thickness, nZ, as additional arguments in the function:

clusteralign_astra_reconstruct(cosine_sample,φ,ψ,aligned_filename,angle_filename, fiterror_filename, bin, nZ)

Dual axis

The two tilt series should be aligned separately using ClusterAlign, and then the supplied Matlab function reconstruct_dual.m handles the alignment between the tilt-series based on a single reconstruction from all available projection images. In order to assist filling up the parameters in the function we offer a Matlab script **run_dual.m**. You just run the script and point to the appropriate output.txt files generated by the two alignments, and it will reconstruct a single 3D image automatically.

5. Optional interface with other software

5.1 IMOD

The fiducial locations file filename.fid.txt can be converted to fiducial models in IMOD. Add the file to the working folder used in the IMOD preparation (see section 3), cd to the folder in Linux and generate the binary .fid file using the IMOD library command:

point2model -in filename.fid.txt -image filename.mrc -open -zero -ou filename.fid

Loading the binary model and testing/ manually modifying the locations works using:

3dmod filename.mrc filename.fid

To move back to ClusterAlign with corrected fid.txt tile the binary can be converted to text via

model2point -object -zero filename.fid filename.fid.txt.

The fid.txt file can be loaded to ClusterAlign in the settings window (‘Load optional fid.txt’). In this case, image analysis in ClusterAlign will be skipped and only a rigid body model will be fitted to the points and the output files will be generated (including aligned stack and optically reconstruction file).

However, to continue working with ETOMO using the fiducial file you would need at least a dummy prealigned stack, since IMOD assumes the points are measured with respect to the prealigned images. To prepare a dummy prealigned file in ETOMO click 'Calculate Cross correlation' 'Generate Coarse Aligned stack' and 'Done'. Then use the following command to translate the points in the binary fid file from the framework of the original stack processed in ClusterAlign to the framework of the prealigned images:

imodtrans -i filename_preali.mrc -2 filename.prexg filename.fid filename.fid

Alternatively, if the fiducial model prepared by ClusterAlign already refers to a prealigned stack you should only care to modify the filename in filename.fid to match exactly with the tilt-series name of the IMOD project. In both cases, it is possible to go directly to Fine Alignment in ETOMO and click 'Compute Alignment' and 'View/Edit Fiducial Model' (that will run 3dmod and show the fiducial locations with respect to the prealined images). It is sometimes needed to correct manually the locations in high tilts, if not enough fiducials are found to track the clusters reliably. Click 'done' and move to 'Final Aligned stack' task and click 'Create full aligned stack' to generate the final ali file. Then you can use the full capabilities of reconstruction in ETOMO/IMOD.

5.2. Tomoalign / Tomorec

The fiducial locations reported by ClusterAlign may be exported to tomoalign for fitting to a model of rotation + warping and to generate aligned file and reconstruction.

The commands in Linux are read as follows if the axis of rotation is along x axis ("r 90" in tomoalign denotes the orientation of the rotation axis is 90 deg compared to y axis):

tomoalign -i filename.nogaps.fid.txt -a filename.rawtlt -o alignment_param -t thick -A mrt -r 90

Tomoalign provides a detailed report of the residual errors in the generated fiducial model. The parameter files, however, can be used only with tomorec for reconstruction, e.g.:

tomorec -a alignment_param -o filename.rec.mrc -i filename.mrc -x <Nx> -y <Ny> -z <Nz> -w sirt -l 30

It is possible to view the reconstruction using 3dmod after the file is resliced.

5.3 Tomo3d/ IMOD-tilt reconstruction

The normal _ali.mrc file generated in ClusterAlign (after ticking the checkbox 'Export normal-ali file for Reconstruction') is prepared to match the standard ali file used in IMOD, tomo3d and other software. This aligned stack is rotated in the XY plane to a vertical rotation axis. Then a weighted back projection reconstruction can be produced by the IMOD function

submfg tilt.com

Alternatively, using functions from tomo3d, IMOD and Bsoft use the following command lines in Linux to view the 3D reconstruction:

newstack -in filename_ali.mrc –ou filename_ali.shr2.mrc -shrink 2 -antialias 5

bnorm -ver 7 -type simple filename_ali.shr2.mrc filename_ali.shr2.bnormS.mrc

tomo3d -i filename_ali.shr2.bnormS.mrc -a rawtlt -z 512 –o filename.WBP.mrc

3dmod filename.WBP.mrc

5.4 Assessing alignment in Matlab with Astra toolbox

The program clusteralign_test.m estimates reconstruction errors based on misalignment between the original projections and reprojection of the reconstructed 3D stack based on back projection algorithm.