NanoJ-Core User Manual

Tested with NanoJ-Core version 2.1RC1 October 2018

Written by Siân Culley, Henriques Lab @ MRC-LMCB, UCL

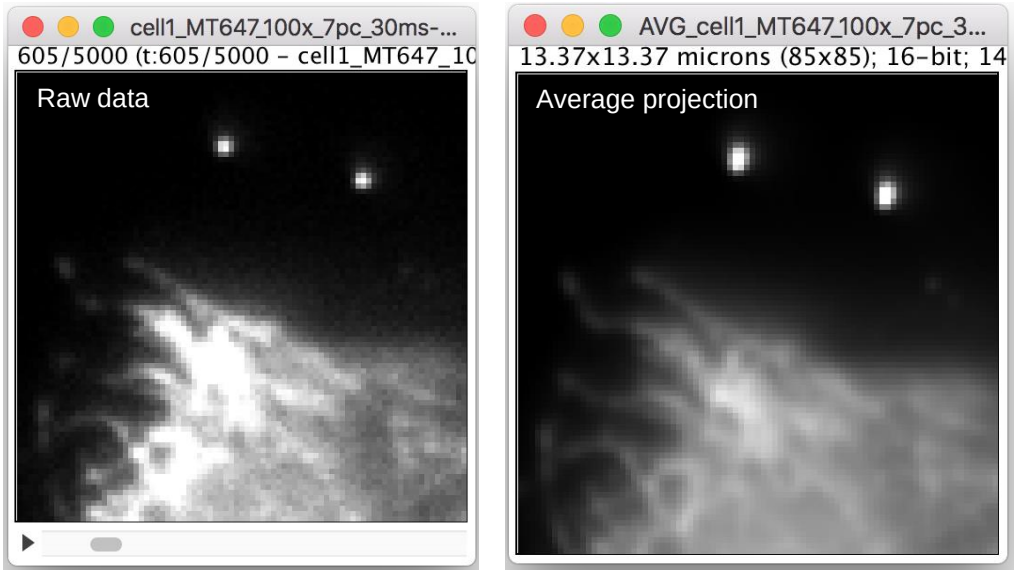
Table of Contents

INSTALLATION

Installation guides for NanoJ-Core can be found at: https://github.com/HenriquesLab/NanoJ-Core. Please follow the installation instructions before proceeding with anything else in this manual!

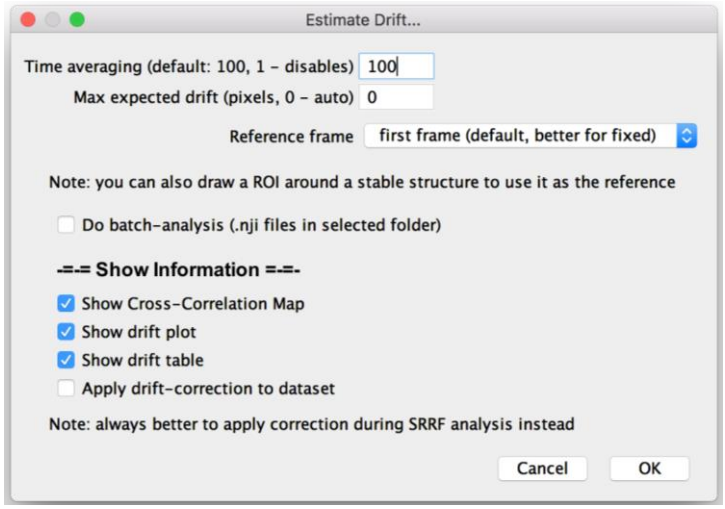
Drift Correction Menu

NanoJ-Core allows you to generate a drift correction table that can be used to correct drift in unprocessed raw data or during SRRF analysis. To demonstrate this I will be using a 5000-frame dataset that also contains fiducial markers.



Estimate Drift

This must be run to generate the 'drift table' which is later used to correct drift. Selecting the 'Estimate Drift' option from the 'Drift Correction' menu will bring up the following dialog box:



Time Averaging

This sets the number of frames to average together to make coarser timepoints on which the cross correlation analysis will be run to calculate drift. Setting this value to 1 will calculate straight frame-to-frame cross correlations and while this should capture drift very accurately, it will also be very susceptible to noise. Conversely, setting this value high will average out noise but will also give lower sample of the drift (which is then interpolated). As a general rule of thumb, less averaging is needed for datasets where the image content is similar in each frame (e.g. widefield acquisitions) and more averaging is needed for datasets where the image

content is different in each frame (e.g. single molecule localization datasets with blinking molecules). As a rough estimate, we normally use 5-10% of the total number of frames as the number of frames to average.

Max Expected Drift

This refers to the maximum expected drift between the first frame of the dataset and the last frame of the dataset in units of pixels. Setting this to 0 will allow the algorithm to automatically determine the drift without any limitations. It is only really worth changing this value from 0 if running the algorithm gives clearly incorrect results with large jumps in estimated drift.

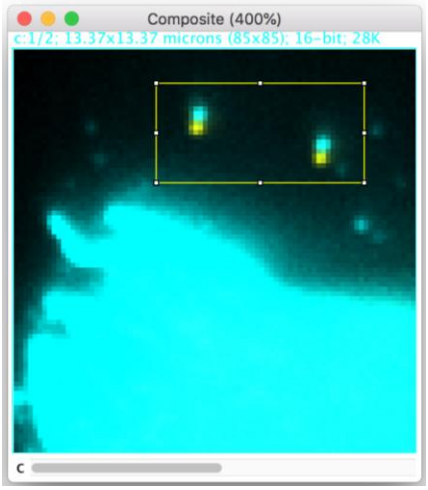
Reference frame

If this is set to 'first frame (default, better for fixed)' then every averaged group of frames will be compared to the first average group of frames to calculate drift. If this is set to 'previous frame (better for live)' then every averaged group of frames will be compared to the previous averaged group of frames (e.g. group 2 vs group 1, group 3 vs group 2). For static samples, it is best to compare to the first frame, and for live samples where there may be slow scale drift overlaying the faster scale sample motion it is better to compare to the previous frame.

Fiducial markers

For acquisition of blinking single molecule localization microscopy datasets, it is useful to include fiducial markers within the sample (e.g. beads evaporated onto the coverslip which do not bleach over the course of the acquisition). If you have fiducial markers within your field of view, you can draw a ROI around them and the drift correction algorithm will only use this region to estimate the drift.

When drawing an ROI around fiducial markers, ensure that the fiducial markers are contained within the ROI throughout the whole duration of the dataset. For example, in the test data set I am using a ROI of the size shown on the frame below – I have overlaid the first frame (cyan) and last frame (yellow) to demonstrate the fiducial markers are fully contained within the ROI.

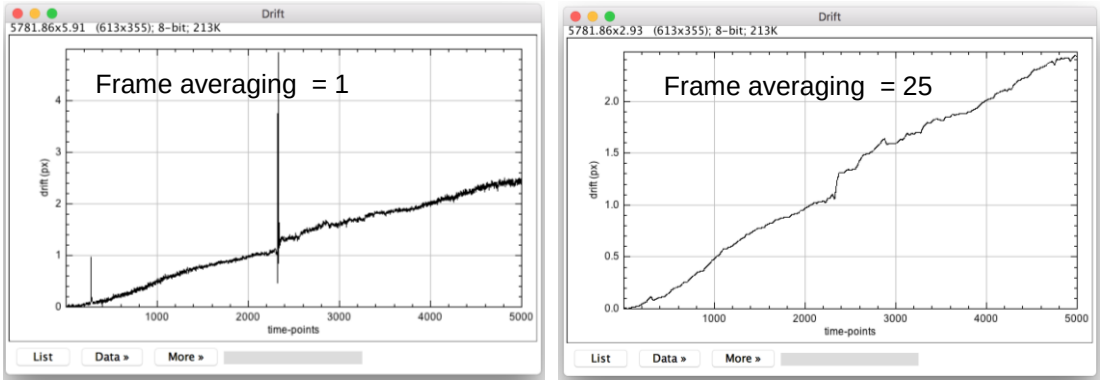


Do batch-analysis

If you have a folder containing multiple raw datasets saved in the .nji format, drift estimation will be performed on each of these datasets with the same settings for each of the above options each time. One drift table will be created for each dataset.

Show information

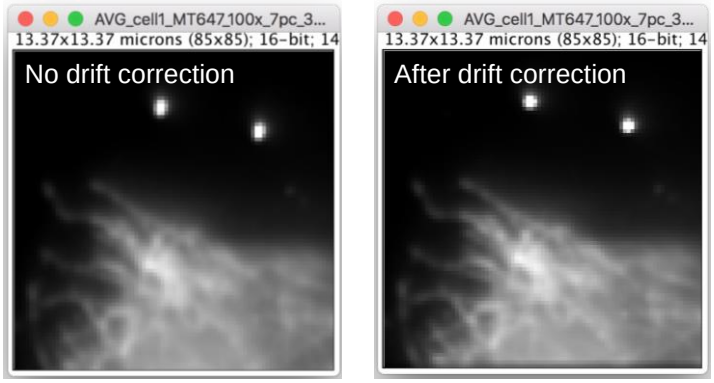
The options here allow you to check the cross correlation maps and estimated drift values. These are useful diagnostic tools for assessing the quality of the drift estimation and optimizing the number of frames to use for averaging. The most useful of these is the drift plot; this should be smooth and free from sharp jumps. This is shown in the example below where there are drift plots for a frame averaging of 1 (i.e. no averaging) compared to frame averaging of 25. In this case the 25 frame-averaged drift estimation smooths out single-frame jumps which are prevalent when no averaging is used.



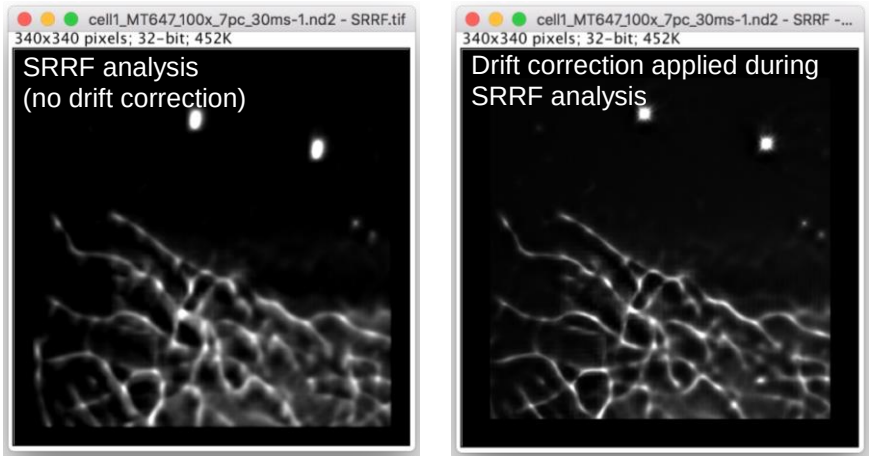
If you cannot eliminate sharp jumps in the drift plot, then it may be worth examining the troublesome frames in the dataset to see if, for example, there are erroneous blank frames or clear problems with the image such as the microscope having been knocked during the acquisition. If using fixed-cell data it may then be appropriate to remove the responsible frames from the dataset and re-run the drift correction. Obviously this should not be done for live-cell data as it will disrupt to the time series.

Apply drift-correction to dataset

This will create a new image where frames are translated according to the drift table to create a dataset where all frames are aligned. Note that this will introduce some bordering into the translated frames. Below are average projections of the 5000-frame dataset before applying drift correction and after applying drift correction.



If you intend to run SRRF analysis on the dataset, then do not apply the drift correction at this stage. There is an option within the SRRF dialog box to apply the drift correction during analysis, and this gives better results than running SRRF analysis on a pre-corrected dataset.



CORRECT DRIFT

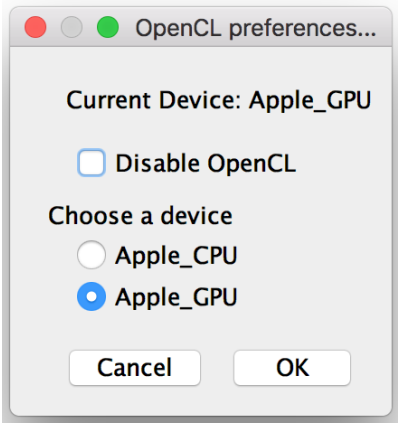
Selecting ‘Correct Drift’ from the menu will ask you to select a saved drift table in the .njt (NanoJ table) format and will apply it to the open dataset. This is the same as checking ‘Apply drift correction to dataset’ in the ‘Estimate Drift’ dialog box. Note that the drift table and dataset to be drift-corrected must both have the same number of frames in order for this to work.

OpenCL Menu

OpenCL is the framework which allows NanoJ to run calculations on the Graphics Processing Unit (GPU).

OpenCL Preferences

Selecting ‘OpenCL preferences’ will bring up the following window:



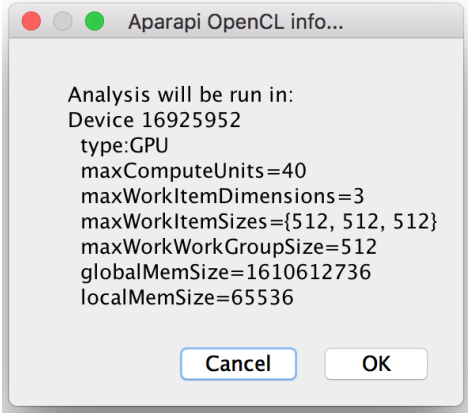
‘Current Device’ indicates the processing unit that NanoJ is currently trying to run calculations on. If NanoJ can find a suitable GPU on your computer then it will automatically select that as the current device, as shown here.

You can disable running calculations on the GPU if you check the ‘Disable OpenCL’; however, this will result in a noticeable decrease in performance of computationally intensive NanoJ algorithms.

Beneath ‘Choose a device’ will be a list of the suitable processing units on your computer for running NanoJ calculations. If you have multiple graphics cards, you will probably want to select the most powerful one. Again, if you select a CPU rather than a GPU here, then calculations will still be parallelized but at lower performance than if you are using a GPU.

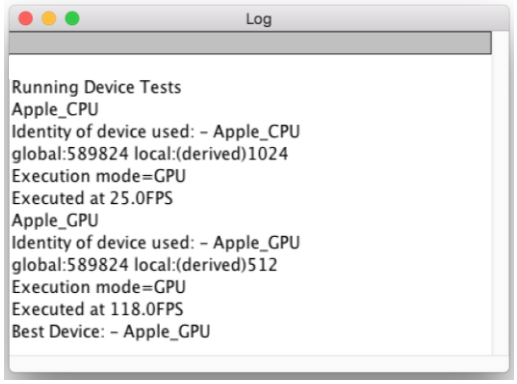
Show OpenCL Info

This will open a window displaying information about the current device that will be used for NanoJ calculations.



TEST OPENCL WITH MANDELBROT

This runs a test (zooming in on the Mandelbrot set of complex numbers) to assess the performance of NanoJ-parallelized computations run on all available OpenCL devices. This brings up a log window suggesting which device should be set as the current device in OpenCL preferences (here Apple-GPU, which offers a 4.7-fold increase in performance compared to Apple-CPU).

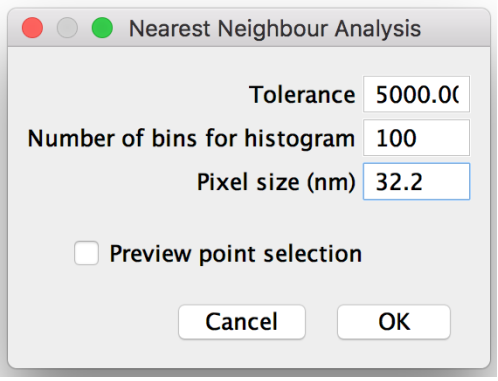


Peak Localisation Menu

This menu contains methods for single-colour and dual-colour nearest-neighbours analysis of datasets containing distributions of point-like patterns.

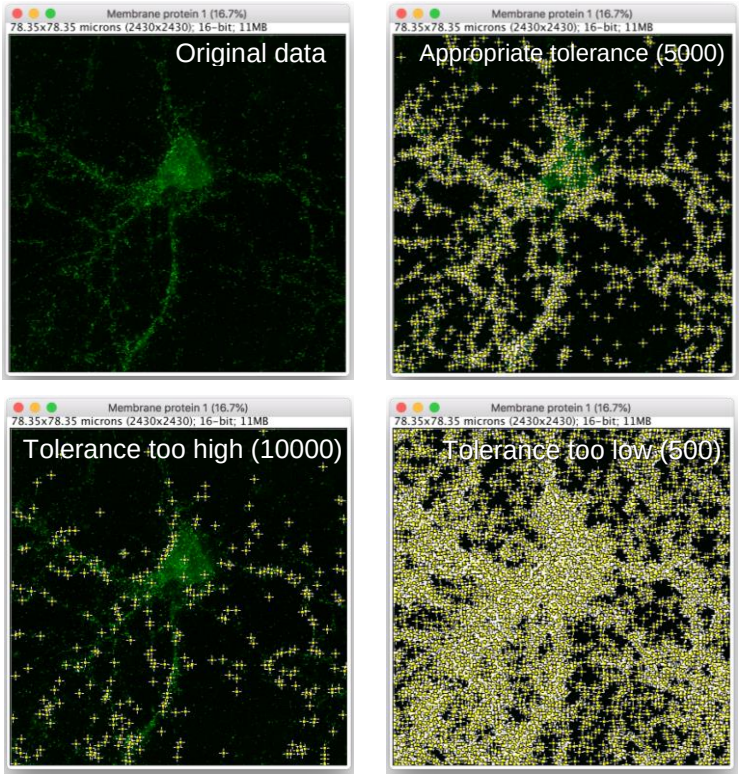
Nearest Neighbour Analysis

This algorithm detects peaks in an image and for each peak finds the distance to its closest neighbour. The dialog box for this is shown below:



Tolerance

This sets the threshold for peak detection. A low tolerance will yield many detected points, while a high tolerance will yield fewer detected points. The correct value will depend on the signal-to-noise ratio of the dataset used.



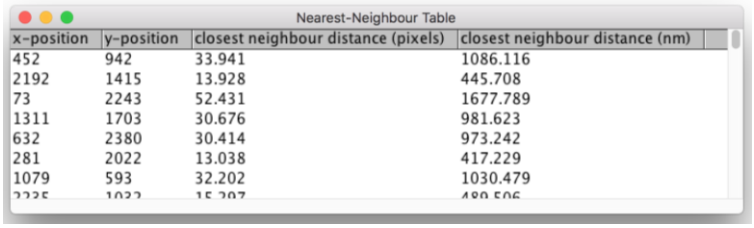
Number of bins for histogram

The algorithm will output a histogram of the nearest neighbor distances in the image; this option allows you to set the number of bins in the histogram (default = 100).

Pixel size (nm)
Set the pixel size of the image to get results in units of nanometers.

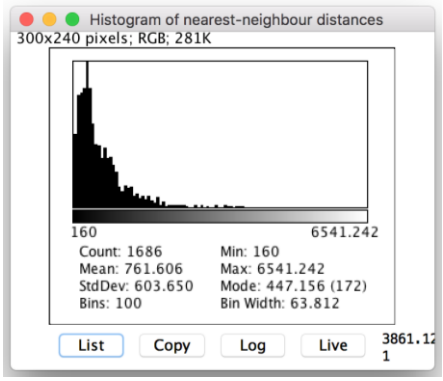
Preview point selection
If this box is checked, the point selection on the image will dynamically update to reflect the tolerance value.

Outputs
There are three outputs of the nearest-neighbours analysis: a results table showing the coordinates of each identified peak and the distance to the closest neighbouring peak, a histogram summary of the data contained within this table, and a Voronoi diagram of the data (explained nicely here: http://datagenetics.com/blog/may12017/index.html).

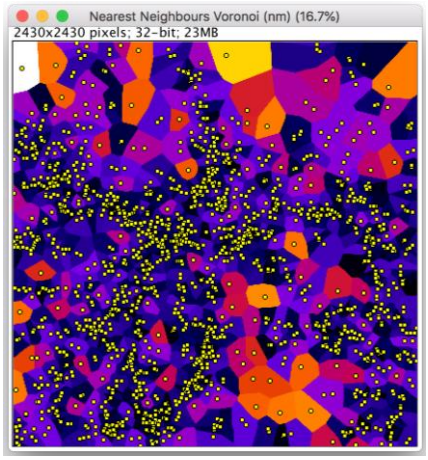


x-position	y-position	closest neighbour distance (pixels)	closest neighbour distance (nm)
452	942	33.941	1086.116
2192	1415	13.928	445.708
73	2243	52.431	1677.789
1311	1703	30.676	981.623
632	2380	30.414	973.242
281	2022	13.038	417.229
1079	593	32.202	1030.479

The ‘x-position’ and ‘y-position’ in this table are in units of pixels.



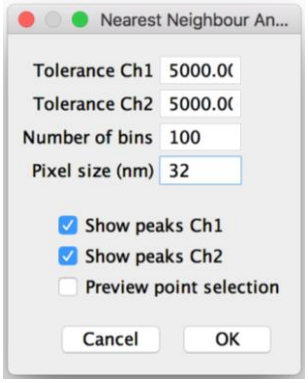
This histogram shows us that the most common separation between adjacent peaks is 447nm, and that separations range from 160-6500nm (note, the original dataset here is a SIM image, hence the sub-diffraction limit smallest separation!).



The Voronoi diagram is overlaid with the detected peaks from the original dataset. The values within the tiles are given in nanometers, and the map is colour-coded using the ‘Fire’ LUT.

Dual Channel Nearest-Neighbours Analysis

This algorithm detects peaks in both channels of a two-colour image and then for every point in Channel 1 calculates the nearest neighbouring peak in Channel 2. This is then repeated such that for every point in Channel 2, the nearest neigbouring peak in Channel 1 is calculated. The dialog box is as follows:



Tolerance Ch1, Ch2

These are the tolerances as explained for the single channel nearest-neighbours analysis section, but now each channel (Ch1 = channel 1, Ch2 = channel 2) has its own tolerance.

Number of bins

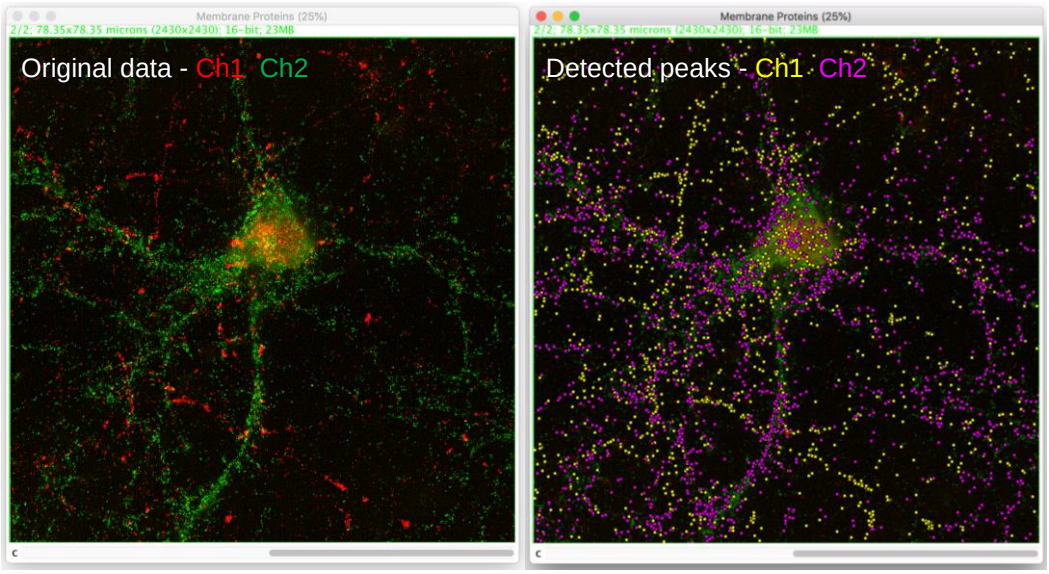
This is the same as explained for the single channel nearest-neighbours analysis.

Pixel size (nm)

This is the same as explained for the single channel nearest-neighbours analysis.

Show peaks Ch1, Ch2

These checkboxes allow you to toggle the detected peaks for each channel on and off when previewing the point selection.



Peaks detected in channel 1 will be shown as yellow point ROIs, and peaks detected in channel 2 will be shown as magenta point ROIs.

PREVIEW POINT SELECTION

If this box is checked, the point selection(s) on the image will dynamically update to reflect the tolerance values.

OUTPUTS

As with the single channel nearest neighbours analysis, the outputs of this algorithm are results tables, histograms and Voronoi diagrams.

Nearest-Neighbour Table: Channel 1 as reference

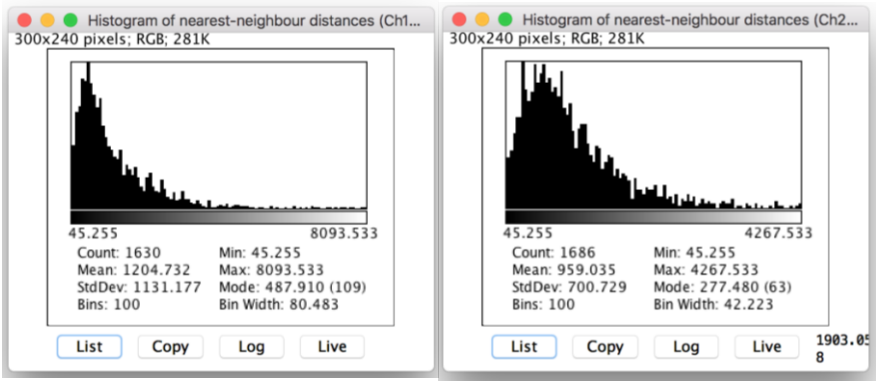
particle index	x-position	y-position	closest neighbour distance	closest neighbour distance (nm)	closest neighbour in Ch2
0	1034	1530	19.925	637.595	848
1	459	1993	80.808	2585.869	1528
2	465	1851	46.872	1499.909	1473
3	1007	1397	5.099	163.169	44
4	1025	1527	13.454	430.516	848
5	1023	1523	15.264	488.459	848
6	252	1179	13.416	429.325	116
7	454	1868	43.046	1377.488	1473
8	1245	519	19.235	615.532	1375
9	1209	551	22.847	731.114	1083

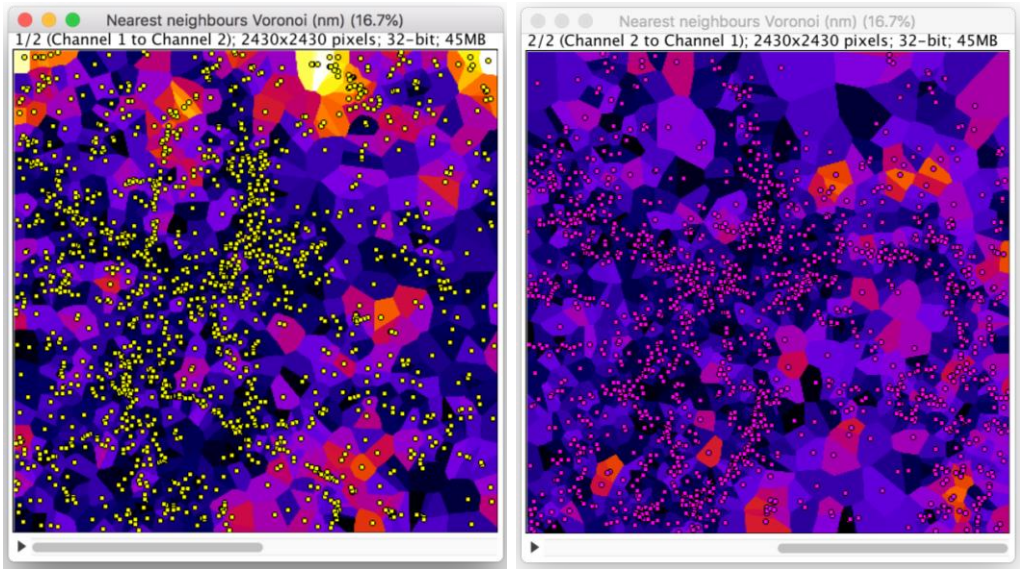
Nearest-Neighbour Table: Channel 2 as reference

particle index	x-position	y-position	closest neighbour distance	closest neighbour distance (nm)	closest neighbour in Ch1
0	452	942	20.616	659.697	118
1	2192	1415	15.811	505.964	564
2	73	2243	27.514	880.436	1387
3	1311	1703	63.127	2020.059	606
4	632	2380	26.000	832.000	718
5	281	2022	23.431	749.784	493
6	1079	593	5.385	172.325	871
7	2235	1032	41.761	1336.359	1229
8	2119	2117	66.611	2131.546	1472
9	2178	1385	34.205	1094.568	564

'Nearest-Neighbour Table: Channel 1 as reference' is a list of every peak found in Channel 1 and the distance to its nearest neighbouring peak in Channel 2. The index of the closest neighbouring peak in Channel 2 is listed in the final column. 'Nearest-Neighbour Table: Channel 2 as reference' is a list of every peak found in Channel 2 and the distance to its nearest neighbouring peak in Channel 1. The index of the closest neighbouring peak in Channel 1 is listed in the final column.

Two histograms are generated: 'Histogram of nearest-neighbour distances (Ch1 -> Ch2)' corresponds to the values in 'Nearest-Neighbour Table: Channel 1 as reference' and 'Histogram of nearest-neighbour distances (Ch2 -> Ch1)' corresponds to the values in 'Nearest-Neighbour Table: Channel 2 as reference'. The distances in these histograms are listed in nanometers.





Two Voronoi diagrams are produced and are output as an image stack. The first image is the Voronoi diagram of distances from Channel 1 peaks (overlaid in yellow) to their closest Channel 2 peak (i.e. the data in 'Nearest-Neighbour Table: Channel 1 as reference'). The second image is the Voronoi diagram of distances from Channel 2 peaks (overlaid in magenta) to their closest Channel 1 peak (i.e. the data in 'Nearest-Neighbour Table: Channel 2 as reference'). The values within the tiles are given in nanometers, and the map is colour-coded using the 'Fire' LUT.

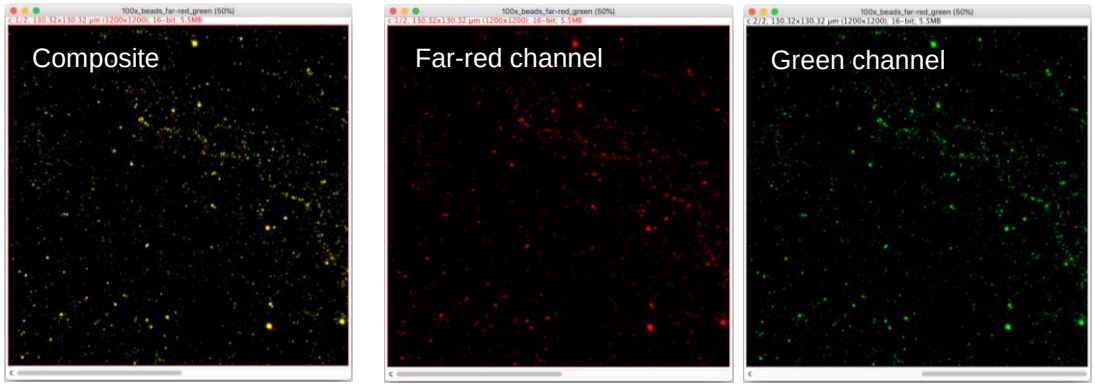
Registration Menu

The channel alignment functionality of NanoJ allows for the estimation of a correction map that can be used to correct for aberrations between two different optical paths. This is most commonly used for correcting for chromatic aberrations between different imaging channels on the same microscope, although we have also shown that it can be used for correlative imaging of the same field of view between different microscopes (https://doi.org/10.1038/nmeth.4605). In this manual we will focus on the application to chromatic correction only.

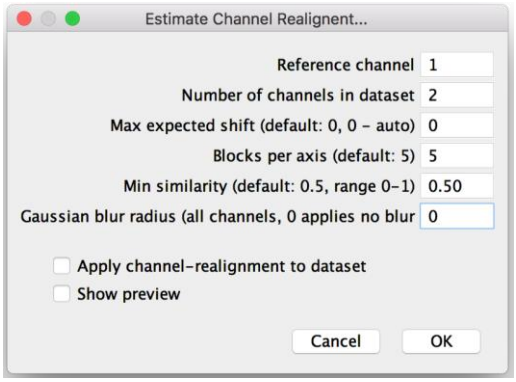
In order to get the best out of the NanoJ channel realignment tool, you should acquire a calibration dataset using the same optical paths as would be used for the actual data acquisition (same objective, additional magnification, filter sets etc). The test sample for this calibration data should be a coverslip coated in fluorescent beads that emit in the wavelengths relevant for your experiment (a good preparation protocol can be found in https://doi.org/10.1038/nprot.2017.019). The imaged region of the calibration sample should have many beads covering all parts of the field-of-view for the most accurate results.

Register Channels – Estimate

Open your calibration dataset as a multi-channel image stack. Here, I have acquired a field of 100nm TetraSpeck beads in the far-red channel (slice 1) and the green channel (slice 2).



With your calibration data open, select ‘Realign Channels – Estimate’ from the Channel Alignment menu in NanoJ to bring up the below dialog box.



Reference channel

This denotes the channel that will be used as the template to realign the other channels on to. We recommend that this is a channel with good signal-to-noise ratio without any additional background structure etc. Here, I have selected channel 1 (far-red).

'Number of channels in dataset' should be set to the number of different channels in your stack; in this case, there are 2 (far-red and green).

Max expected shift

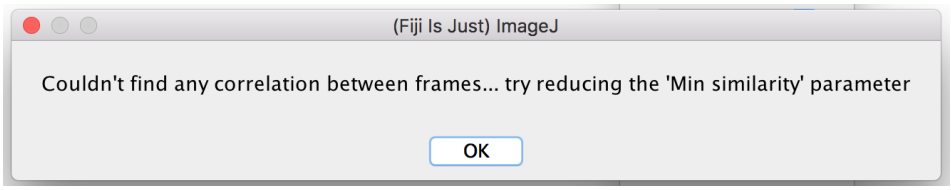
This refers to the maximum expected misalignment between the two channels in units of pixels. Setting this to 0 will allow the algorithm to automatically determine the shifts without any limitations. It is only really worth changing this value from 0 if running the algorithm gives clearly incorrect results with too-large shifts.

Blocks per axis

This indicates the number of blocks in each dimension over which the shift will be calculated. If your calibration image does not contain many beads, this value should be set to a lower value to avoid the algorithm trying to perform realignment on blocks containing no structure.

Min similarity

This denotes the threshold for accepting the cross-correlation value for a block as sufficient to assign a realignment correction. We have observed that setting this value to 0.5 tends to give the most robust results. Low cross-correlation values will be returned if there is, for example, low signal-to-noise ratio in the channels. You will need to reduce the value of 'Min similarity' if running the algorithm returns the following dialog box:



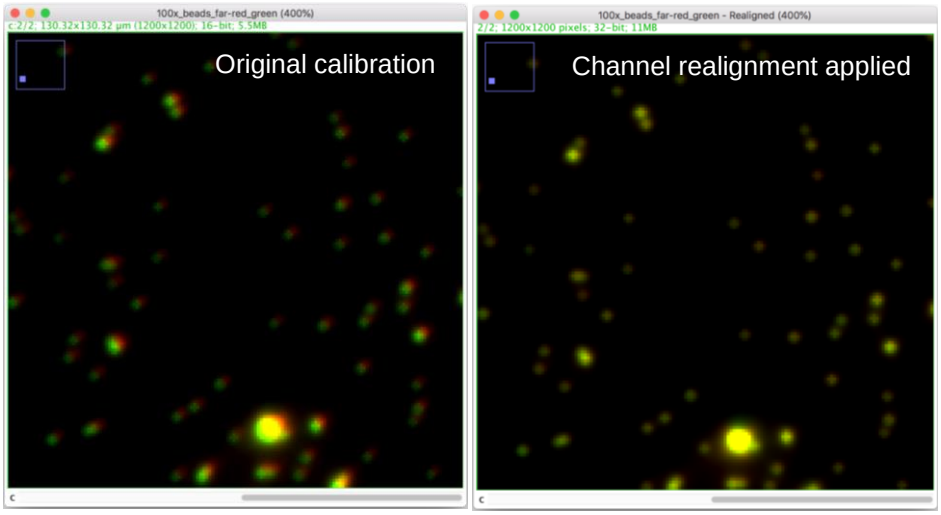
However, if this is happening repeatedly then it may be worth checking the quality of the calibration sample that you are using (i.e. sufficient coverage of beads, good signal in both channels…).

Gaussian blur radius

This allows you to apply a blur to the image before running the cross-correlation based realignment. Again, this can be helpful if the calibration images are noisy.

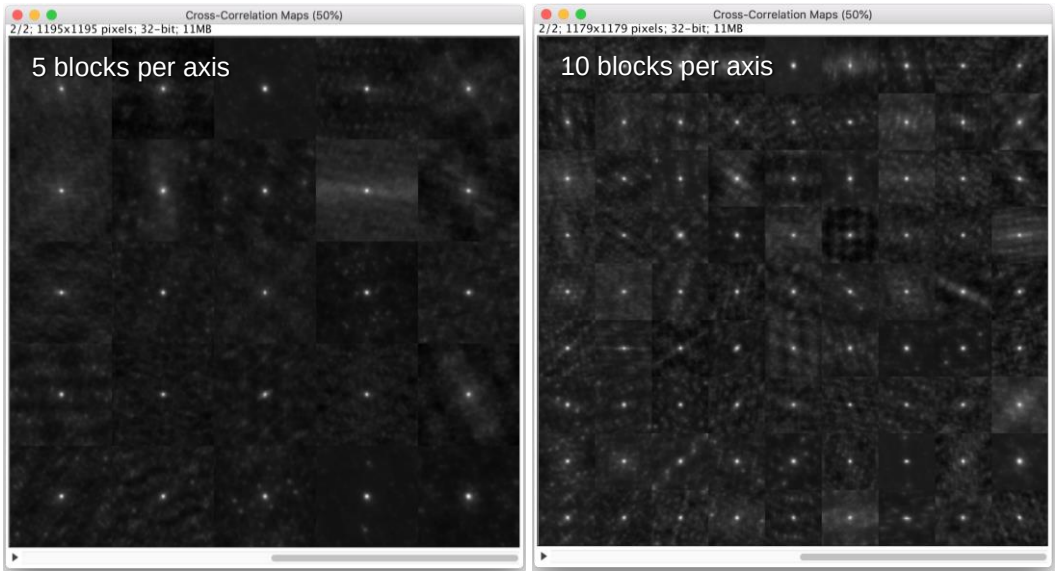
Apply channel-realignment to dataset

This option, if ticked, will generate a realigned version of the calibration dataset after the cross-correlation analysis has been performed. This is a useful way of validating if the realignment has performed well.



Show preview

This will, when checked, run the realignment algorithm and display the calculated translation mask, realigned calibration data, and block-wise cross-correlation maps. These three images will then update in real time as values are changed in the dialog box. This can be useful for determining e.g. the number of blocks to use and the min similarity. The cross-correlation maps are shown below for two different numbers of blocks per axis (5 and 10 respectively). Hovering the mouse over the central peak of each block will show the value that is subject to the 'min similarity' threshold in the Fiji window. The cross-correlation maps will have the same number of channels as the calibration dataset, and will show the local cross-correlation of that channel against the reference channel (the slice containing the reference channel is a cross-correlation against itself and should be ignored).



Translation masks

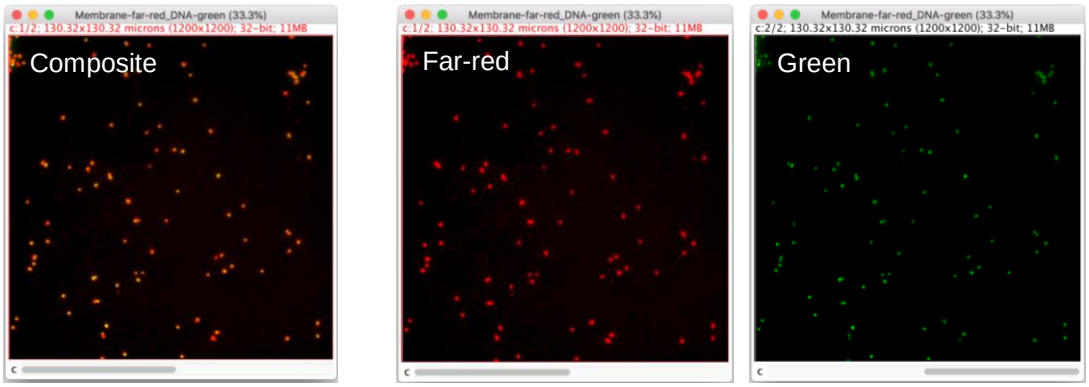
Press 'OK' in the dialog box to generate the translation mask. The translation mask has the same number of channels as the calibration dataset, and is twice the width of the original dataset. The left half is the interpolated offset map in x for that channel against the reference, and the right half is the interpolated offset map in y for that channel against the reference. Again, the slice corresponding to the reference channel itself should be 0 everywhere. You will need to save this translation mask stack to be able to later apply it to your data.

Register Channels - Apply

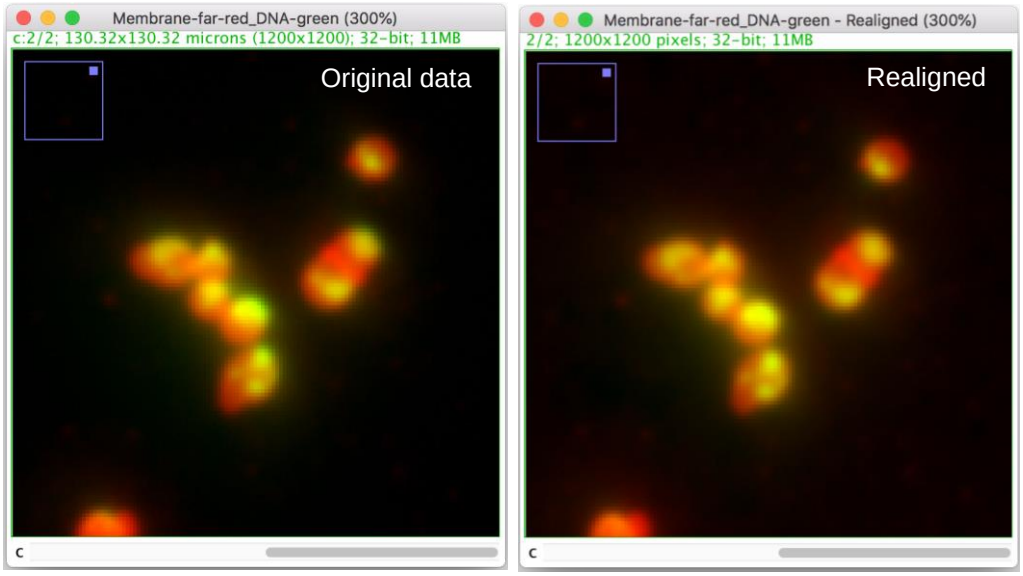
Open your data that you wish to apply the realignment to. This 'target' data must have the same dimensions as the calibration dataset that was used to generate the translation masks. If the field of view was cropped on the microscope to acquire the target data then the calibration

dataset must be cropped to the exact same area in the field of view before being run through the channel realignment estimation step to generate corresponding translation masks.

The target dataset must also have the same number of channels present as the translation mask, and in the same order. To demonstrate this, I will be using a target data set of *Sulfolobus* cells stained with a far-red membrane dye (slice 1) and a green DNA marker (slice 2). These images were obtained using the same optical paths as the calibration dataset.



Selecting 'Realign Channels – Apply' with bring up a dialog box asking you to select the file corresponding to the translation mask generated during estimation.



As shown above, applying the channel realignment to this datasets moves DNA that appeared to be outside the cell in the original data to back within the cell membrane border.

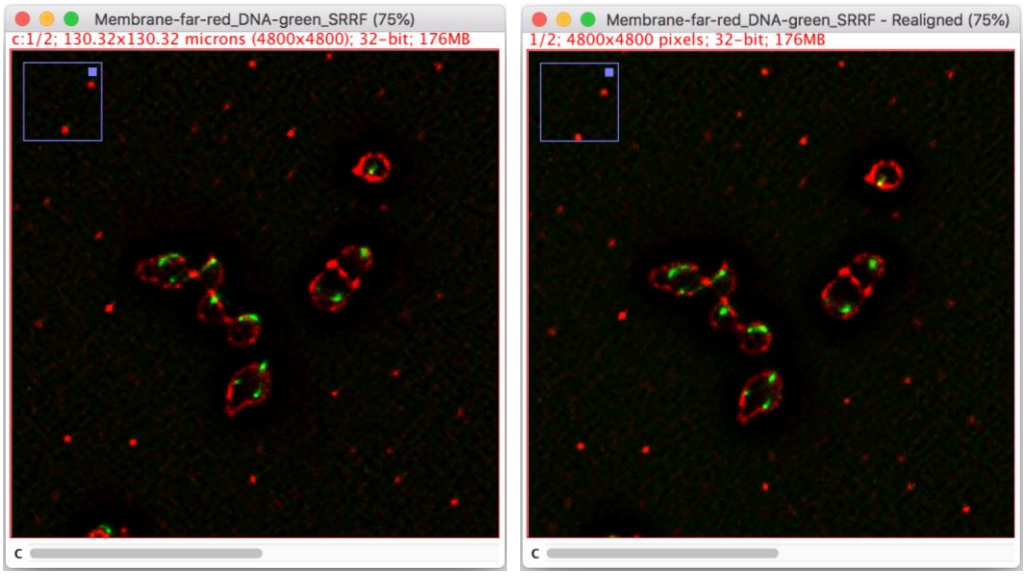
Using Channel Alignment With SRRF Data

You typically cannot apply a generated translation mask directly to a SRRF-analysed dataset as the SRRF images are usually different dimensions to the raw data (due to the magnification applied in SRRF analysis). Therefore, you need to appropriately scale the translation mask if you want to apply it to data analysed with SRRF.

First, you need to scale the dimensions of the translation mask by the same magnification factor as used in SRRF. This will ensure that the size of the translation mask matches the size of the SRRF image. You can do this by navigating to the 'Image' menu in Fiji and selecting 'Scale…' while the translation mask image is open and active. Secondly, you need to multiply the shift

values contained within the translation mask by the magnification to ensure that the corrections themselves are being applied with the correct magnitude. You can do this by navigating to the ‘Process’ menu in Fiji, then going to ‘Math’ and selecting ‘Multiply…’. Once you have scaled and multiplied the translation mask, save a copy and then you can apply this to the SRRF image as described in ‘Realign Channels – Apply’.

The below image shows the effect of this on the same dataset as used in ‘Realign Channels – Apply’ but after it had been analysed using SRRF.



NanoJ file formats

Batch processing functions in NanoJ require the data to be saved as the NanoJ Image (.nji) file format. This is to prevent unexpected behaviour arising from e.g. proprietary microscope file formats which typically need to be imported using the Bio-Formats importer. Similarly, data tables for input and output in NanoJ (most commonly encountered in the drift correction analysis) require the NanoJ Table (.njt) file format.

.nji and .njt files are just zipped tiff files, and as such take up less space than the native format files. Also, you can just replace the .nij/.njt extension and replace it with .zip if you wanted to open them elsewhere.

The naming convention for .nji files is [image title]-00[n].nji. For images that fit within a single .nji file, 'n' will be 0 and only one file will be created. Larger files will be saved as a series of .nji files with increasing values of n (e.g. Image-000.nji, Image-001.nji…).

Open NanoJ Image (NJI)…

This brings up a dialog box to open an image currently saved in the .nji file format. When opening an image that is spread over several .nji files it does not matter which .nji file you open, the whole dataset will still be read.

Save image as NJI…

This brings up a dialog box to save the current selected image in the .nji file format.

Convert image sequence files to NJIs…

This function converts all files with a given extension in a selected directory into .nji files. It does not delete the original files, just creates additional copies in the .nji format that can then be used in NanoJ batch processing. The .nji files are saved in the same location as originally selected.

The supported file extensions that can be converted are : .tif, .czi, .nd2, .vsi (select the relevant file extension from the dialog box that appears after you choose the directory containing the files).

Open NanoJ Table (NJT)…

This brings up a dialog box to open a table currently saved in the .njt file format and opens it in a 'Results'-type window. Metadata will appear in the Log window.

Save Results-Table as NJT…

This brings up a dialog box to save the current selected Results window in the .njt file format.

Convert Results-Table file (.xls) to NJTs…

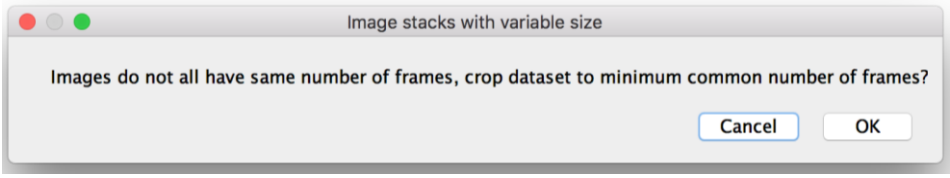
This function converts all .xls files in a selected directory into .njt files. It does not delete the original files, just creates additional copies in the .njt format that can then be used in NanoJ batch processing. The .njt files are saved in the same location as originally selected.

Set Compression (NJI & NJT)…

This lets you set the compression level of the zipping process to make the .nji and .njt files. The default value for this is 3 (maximum of 10).

Open image files in folder as image sequence…

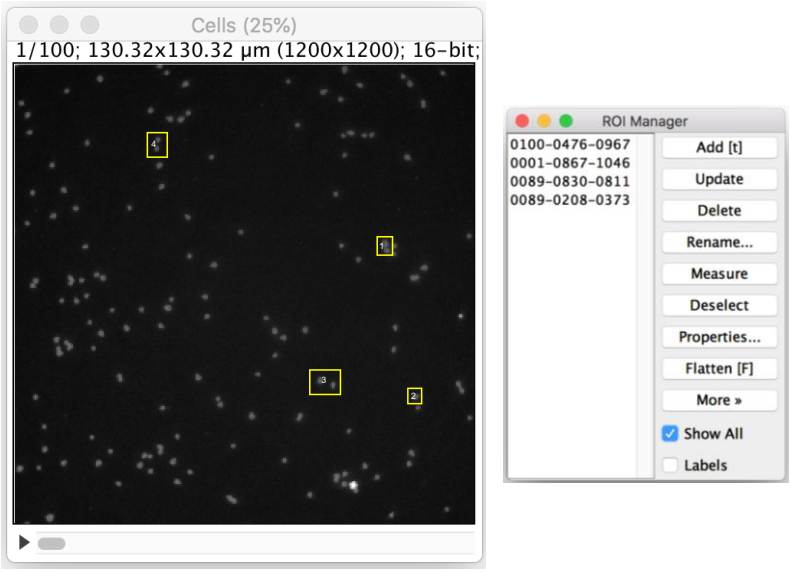
This function will concatenate all image files with a given extension (supported: .tif, .czi, .nd2, .nji, .vsi) into a single image sequence and open it. These images must all have the same (x,y) dimensions. We find this particularly useful when doing multidimensional SRRF analysis, as often each burst of ~100 raw images required for a single SRRF image is saved as an individual image file in some of our acquisition protocols. You may see the following message when running this function:



If you want to ensure that all of the individual images comprising the final stack have the same number of frames then press ‘OK’ in this dialog box, and datasets with additional frames will have frames deleted from the end until a common size can be reached. For example, if you are opening 5 files with the following numbers of frames: 100, 100, 99, 100, 99 then pressing ‘OK’ will remove the last frame from the first, second and fourth datasets such that they all have 99 frames prior to concatenation. The final dataset will thus have 495 frames. Pressing ‘Cancel’ will not remove any frames prior to concatenation; in this case the final dataset will have 498 frames.

Extract ROIs and save as NJIs…

This option allows to you save subregions of an image stack as separate files. Draw rectangular ROIs onto your image and add them to the ROI Manager. The ROIs do not have to be actively displayed on the image when you run this command; they just have to be within the ROI manager.



The extracted image ROIs will be image stacks with the same number of frames as the original dataset. They will be saved as .nji files in the selected directory as with ‘_ROI[n]’ appended to the image title (in this case: Cells_ROI0-000.nji, Cells_ROI1-000.nji, Cells_ROI2-000.nji, Cells_ROI3-000.nji). Note that this numbering convention is zero-indexed, and so will correspond to the ROI label in the ROI manager that is one higher. I recommend also saving the ROIs that you used for extraction so that you can trace back extracted images at a later

date if necessary. To save the ROIs within the ROI Manager as a zipped set, make sure that no ROIs are selected within the ROI Manager window (i.e. nothing is highlighted) and then press ‘More >>’ and then ‘Save…’.

STOP NANOJ EXECUTION…

If you want to cancel any NanoJ process that is currently running (this can also be in other packages such as SRRF and SQUIRREL), then press this!

RESET PREFERENCES

All NanoJ functions which have user-definable parameters have an associated preferences file that remembers the last-used settings. Pressing this will prompt you to immediately restart ImageJ without taking any further action; upon restarting ImageJ all NanoJ variables will have been reset to their default values.

NANOJ INFORMATION

SHOW MTA

This displays the Material Transfer Agreement for the NanoJ software.

NANOJ CORE VERSION

This displays the current release version of NanoJ-Core that you are using.

NANOJ-CORE WHAT’S NEW

This displays new features that have been added in the most recent release of NanoJ-Core.

NANOJ-CORE USER MANUAL AND PAPER

This will bring up links and QR codes to get to the manual (and thus enter a circular loop by finding yourself here again!) and the most recent version of the manuscript.