Imagej manual stitching

Continue

Image stitching is a major problem in modern microscopy. While microscopes continue to improve in terms of speed and field of view, the fact remains that tissue and organism level imaging requires acquisition of multiple multidimensional fields of view. Powerful open source optimized tools exist for image stitching (e.g. Fiji's grid/collection stitching) but they are focused on whole image correlation which often fails in the context of complicated and noisy images. Upon failure of correlation, images are often dramatically misaligned, leading to gaps in the stitched images and inappropriately positioned tiles. In addition, many of these tools perform stitching from the hard disk. While this is desirable for conservation of RAM, large scale stitching becomes very slow. In reality, stitching is best performed on a subset or processed version of the full 3D multi-tile data set. For example, one can determine xy stitching positions from max projections of a subset of the data or single carefully chosen channels or slices of the data. This stitching can be performed rapidly, followed by a large scale stitching of the entire data set from those coordinates. We have attempted to address these complicated challenges in number of different ways with open source tools available on github: github.com/jayunruh (in Jay_Plugins, jalgs.stitching). Firstly, we attempted to address the poor success rate of image correlation by borrowing a trick from spatiotemporal image correlation (STICS) or optic flow. Instead of correlating neighboring images in a image-wide fashion, we correlate sub-images spaced along the shared seam between the neighboring images. These small focused overlap regions show enhanced correlation where possible and combined analysis (e.g. via projection) of these sub-images selects for regions of strong overlap without spurious correlation from non-overlapping regions. This methodology has the added effect of trading a small number of large image correlations for a larger number of small image correlations. This significantly reduces the time required for the correlation and alignment process. Secondly, we have changed the workflow for the stitching process. Given the desire for stitching of only image subsets, we have split the stitching process into two parts. Image subsets (single slices or projections of selected channels) are selected for coordinate optimization via the image correlation process described above. This process generates an optimized xy coordinate set which can then be used for large scale stitching of full 3D tile sets. Coordinate Optimization/Generation Preliminaries: You need an updated copy of Fiji following the Stowers plugin update site. All of the plugins used below are in Plugins>Stowers>Jay_Unruh. Please use the plugins search tool (Plugins>Shortcuts>List Commands) to find the exact location within that menu. To start the process, you need to have a set of image tiles loaded as a time lapse data set (each frame has a subsequent tile). Obviously, this can be a bit tricky to generate, but I will leave that to your general Fiji expertise. One option is to load all of the tiles in order (with nothing else loaded) and then run "merge all stacks jru v1". Secondly, if your data is not collected in a standard configuration (snake by rows or raster by rows starting at one of the corners), you will need to have a position plot with the guessed tile positions. This can either be in real or pixel units. If you have two columns (x and y) of coordinates in a spreadsheet program, you can copy them and paste them into a plot (no column labels with the "text2plot jru v1" plugin. Just select Clipboard from the windows menu and check the X Vals Column? option. Alternatively, you can load the coordinate list from a csv file and plot the columns with "plot columns jru v1". I have put a simple test tile stack here with its coordinates here for testing purposes: Tile Stack Coordinates Plot The first stitching plugin will only search in two dimensions to find the xy coordinate offsets. It is important to select the channel and z slice with the greatest amount of overlap between it's neighboring images for the stitching process. In some cases, the transmitted light image will fit this criteria. In other cases, you may want to perform a max or sum (or min for transmitted light) projection over a number of slices to generate a 2D data set with good overlap. In either case, the currently selected slice and channel will be used for initial coordinate optimization. To perform the coordinate optimization step, run "stitch stics jru v1". Select the tile time lapse and the position plot. If you don't have a coordinate plot but do have snake by rows or raster by rows organization, select "null" for the position plot selection. Next, you will see a set of options for correlation analysis. If "Find Shifts?" is not selected, stitching will be done with the default initial positions. If the optimized coordinates are more than "Median Shift Tolerance" percentage away from the median horizontal and vertical shifts, they will be set to those values. In addition, if the phase correlation is below the correlation threshold, the median shifts will be used instead. Output correlation gives you the option of viewing the projected correlation images. Visually examining these for peaks can be useful for troubleshooting. Finally you can set the criteria for finding the best correlation peak. Highest Crosscorr is typically the best option. If you loaded a coordinate plot, the plugin will ask you if that plot has units (or is in pixels). The default pixel size is derived from the image calibration. If you didn't select a plot, you will need to enter the overlap percentage, the tile dimensions and the tile scan configuration. Once the stitching optimization process starts, a second plot of the initialized stitching coordinates will appear. This plot will be in pixel units. The log window follows the progress of the stitching. First, it lists the top three peaks found in each correlation image and which peak was chosen. Note that the image pairs start with 0, so the first slice is labeled "0". Next, it lists the image pair chosen peaks again and then median horizontal and vertical image shifts. If there are too few images with significant correlation within the median shift tolerance, the median shift will be set to the guess value. Next the image pairs are listed again with individual shift coordinates set to the median values if they don't meet the criteria. Finally, the chosen fixed frame is listed and the average horizontal and vertical overlaps. Upon completion, you will see a second (blue) set of coordinates in the "Stitching_Coordinates" plot. These are the optimized coordinates. If you chose to output correlation images, you will see that stack with slice names showing the image pairs. Finally you will see the stitched image for the selected slice and channel. Stitching with pre-optimized coordinates Once the coordinates are optimized (could be by the method above or by another method), you can use them to stitch large 3D multicolor volumes together. Just like before, you will need to have a timelapse with the frames corresponding to the image tiles. You will also need your coordinate plot. If you use the optimized coordinates from the proceedure above, use the "Select+" button on the plot to select the optimized coordinates (the blue line should change to blue squares). If you want to run this from a macro, run the "PlotWindow Extensions jru v1" plugin at the beginning of your macro and then run Ext.selectSeries(1) to select the appropriate series. Then run "stitch mosaic image jru v1". Select the image and plot as before. If your plot is in pixel units (it should be from the process above), select the ignore scaling option. The stitching process will then proceed and the resulting stitched image will appear on completion. The content of this page has not been vetted since shifting away from MediaWiki. If you'd like to help, check out the how to help guide! This tutorial describes' how to produce an image stack (or 3D image) from an input sequence of tiles' using the Fiji plugins for stitching and registration. Given the origin of the images used in this tutorial, the transformation between tiles can be modeled as a pure translation to generate the mosaic (of a slice). The transformation between slices can also be modeled as pure translation. In the initial setup we need to ensure: All the images are stored in the same folder and have an intuitive sequencing name (such as Tile_X{xxx}_Y{yyy}_Z{zzz}.png), more information below. All slices of the stack are formed by a rectangular grid (NxM tiles) of a equally sized tiles. In our test case, we have 19,600 images, i.e. 140 sections of 10x14 tiles: Stitching As a first step, we start Fiji and go to Plugins › Stitching › Stitch Sequence of Grids of Images: Then, the next dialog pops up to choose the stitching parameters: Here we have to set some important parameters (the ones we don't mention can be left with their default values): Grid size in X, Y and Z: number of tiles per column, row and section. In our case: 10, 14 and 140. The input directory: the folder containing the image sequence. Notice here that you can drag and drop the folder into the text box and the path will be copied. The file names: the template of the file names. The Stitching plugin accept two types of templates, one with a specific number for each x, y or z position, and one with an absolute number (i). In this case, our files will have the template Tile_Z{zzz}_Y{yyy}_X{xxx}.png. That means the file corresponding to the first section, first column, first row, will be called Tile_Z{001}_Y{001}_X{001}.png. The output directory: the folder to store the resulting stitched images. The fusion method: the way the plugin will treat the overlapping areas. In our test case we won't fusion the images, so we select None. The regression threshold: the number below with the plugin won't accept correlation values as valid. The default value (0.3) is quite low, but in our test case works perfect. We then click on OK and the stitching will take place. The plugin will display all intermediate results until the whole sequence has been stitched. After processing the last grid/section, the plugin will display the following message: As result, the stitched images are stored in the output folder: Performance: The stitching of the 19,600 images (732x732 pixels each) took around 117 minutes in a Intel Core Duo at 3GHz, 4GB of RAM, running on Linux 64-bit. Alignment For the alignment of the stitched slices we will use the plugin Register Virtual Stack Slices, under Plugins › Registration: As before, a dialog will pop up where we have to choose the registration parameters: The relevant parameters are: The source directory: we set it to the stitching output folder so the plugins will align the stitched images. The output directory: we set it to the folder where we want to store the final aligned slices. The feature extraction and registration models: in this case, and as mentioned before, we only need a translation to model the transformation. And we click on the first two check-boxes: Advanced setup, to specify the feature extraction parameters. Shrinkage constrain, in other registration models, this guaranties a non-shrinking registration. Since we chose translation, we don't have this problem, but we checked this option anyways to guaranty a complete multi-thread registration. When we click on OK, another dialog pops up to select the Feature extraction parameters: Here, we only increase the steps per octave scale to 5 to find more point candidates to correspondences, and the maximum image size to 1400 pixels, to use more image information. NOTE: these values are usually a good choice, but they can be tuned when working with other types of images. For more details about the parameters, visit the registration plugin page. We then click on OK and the alignment starts. After few minutes (depending on the computer and number of CPUs), when the alignment is done, aligned images are saved to the specified output folder and results will be displayed as a virtual stack. Performance: the alignment of the 140 slices (and their resizing) took around 20 minutes in a 8-CPUs Intel Core Duo at 3.4GHz, 32GB of RAM, running on Linux 64-bit.