Alignment of Electron Microscope Sections

Roger Zou, Julia Ni

July 16, 2014

1 Overview

Alignment of 3D electron microscope (EM) sections with CAJAL-3D API integration. Its use is domain-specific; namely, the program corrects for translations and rotations between image slices. See the README for details about the functions available to the user. The program primarily uses cross-correlation to determine rotation and translation parameters which maximize the correlation between pairwise images. This report provides details on such alignment methods, and extends the aforementioned core technique to provide scalable global alignment of terobytes of EM image data while including additional optimization and error-computation steps for robustness; see below. Future implementation of said method in the connectome estimation pipeline and the Open Connectome Project may also be possibilities.

2 Global Pairwise Alignment

Before an entire 3D image cube can be aligned globally, we must compute alignment between pairs of images using cross-correlation.
Median filtering and histogram equalization, which are performed before sampling in log-polar coordinates in the Fourier domain, detect the shifts in rho and theta (scale and rotation, respectively). This method determines the best rotation required to align pairwise images. Next, we use normal cross-correlation to determine the best translational shift. See the '2D Cross-Correlation' section for more information.
One important step in the cross-correlation process is identifying the transformation parameters from the cross-correlation of two images. The correct transformation between two images is determined by finding the 'correct' peak in their cross-correlation. However, since problems arise when the 'correct' peak is NOT the maximum value, we use a trained support vector machine (SVM) classifier to determine the correct peak among potential candidate peaks. See the 'Peak Detection' section for more information.

2.1 2D Cross-Correlation

Iterate through image stack, and apply the following procedure to each adjacent pair of images.
Alignment with Cross-Correlation Procedure:

1. Apply hamming window to both images.
2. Take the Discrete Fourier Transform (DFT), apply a high-pass filter, and then resample the images in log-polar coordinates.
3. Compute the phase correlation to find the best rho (scaling) and theta (rotation angle).
4. Refine the initial rotation parameter, previously computed from feature matching, by using the theta value from correlation.
5. Rotate the image appropriately, then compute normalized cross-correlation to find translation parameters.
6. Use an SVM to identify the peak in the cross-correlation matrix of images that accurately corresponds to the actual translation parameters.

7. Save the pairwise affine transformation matrix.

The time complexity is $O(nlogn)$, where n is the the number of voxels. This upper bound arises due to the Fast Fourier Transform (FFT) computations. The space complexity is linear to the number of voxels. One optimization step is scaling the image stack before determining image transforms. Scaling between 0.5 and 1 does not appear to have much of an effect on alignment quality while scaling by 0.5 effectively decreases the constant in front of $nlogn$ in the running time by 4.

2.2 Peak Detection

Uses an SVM to classify a local maxima as a peak or non-peak. Picking only the maximum values will not always yield the correct results, as the example on the right demonstrates. The peak is quite obvious, but the maximum values (in red) are clustered elsewhere. The SVM is trained with shape features to distinguish between peaks and non-peaks, and performs the best compared to other methods.

Attempted Methods to find peaks in the cross correlation of two images

1. Identify maximum values. However, this will not always work, as the example on the right shows.
2. Correlate the cross-correlation of two images with a normal distribution. In theory this should accentuate the true peaks, since it takes into account the shape of all possible peaks, but this won't always produce the desired results because artificial peaks may be introduced. Another problem is that this method is quite slow because it has to perform more cross-correlation procedures.
3. Use an SVM trained on five features: number of pixels in image, area of binary thresholded region, maximum gradient, maximum Laplacian, and skewness of the histogram. The cross-correlation images are divided into nine subsections. The maximum is identified in each, evaluated with the SVM, and then the maximum value predicted by the SVM to be a peak is labeled as the actual peak. If no maximum values are predicted by the SVM to be peaks, then no peak exists. This appears be the best method as it accurately identifies peaks which are not the global maxima and its run time is nearly instantaneous.

3 Unused, Functional Code

3.1 RANSAC

Originally implemented to detect folds in the images and subsequently help split images into pieces to aid in alignment process. We have not encountered any thus far, so this functionality was never included in our pipeline.

3.2 SURF Feature Matching

Originally used to correct large rotations before aligning with cross-correlation. Eventually discarded because images are often too different to match features well. Downsampling did not help much for some images.

1. Use Gaussian blur and then downsample (via MATLAB's *imresize*). Also tried median filtering and anisotropic diffusion without characterizable improvement.
2. Detect SURF features (experimented with most feature detectors in MATLAB, but SURF appeared to work the best).
3. Match the features and estimate the rotation parameter.
4. Save the rotation angle and apply the rotation to image for input to 2D cross-correlation step.

Problems: Effectiveness decreases for more dissimilar images. SURF also struggles to identify features or makes spurious matches when there are noise and blemishes in the images. This step does not seem to lead to drastic improvements.
Improvements: Are there ways of using entire cell structure features such as whole mitochondria and other organelles for feature matching? This is how it appears some humans (or at least how I) visually align these EM sections. The current method isn't that useful, but by incorporating these features I feel like it has the potential to be much better.

4 Global Stack Alignment

Uses the pairwise transformation parameters to compute global transformation parameters. Starts with the first image and builds the aligned stack iteratively, taking into account the pairwise transformation between two images as well as the previous transformations of the previous images. Can also reverse alignment given an aligned stack of images, again using the iterative nature of alignment transformations.
Problem: One incorrect transformation causes the rest of the image stack to shift by the same amount. Is this the 'right' thing to allow happen? Also, how to deal with all black images?
Improvements: Other methods of putting all the aligned images together?

4.1 Tune Transformation Parameters

Attempts to improve the estimate of the rotation amount by minimizing an error function. While three different error metrics are available to determine 'accuracy' of alignment (peak-signal-to-noise ratio (PSNR), mean-squared error (MSE), and mean pixel intensity difference.), MSE is usually used. This step is important because log-polar sampling to determine the best rotation parameter is discretized so the 'best' rotation angle may not be available at the cross-correlation step. Here, the algorithm iterates over a small range of rotation and translation parameters surrounding the original estimated parameters, and then chooses the rotation angle and translation shift that minimize the error function.

4.2 Error Minimization

The final (optional) step right now is to use image data outside a specific image pair to align the image pair. This works well by correcting alignments when one pairwise alignment fails for some reason. The specifics of the algorithm are detailed below. The algorithm improves pairwise transformation parameters by computing pairwise transformations from other adjacent images. Different estimates of the transformation parameters between the image pair are computed from solving a linear programming problem. The transformation parameter that minimizes a particular error function is selected.

Problems: None that I know of :-)
Improvements: Other optimization parameters to add? Other constraints that can help to minimize error? Can't think of anything specific at the moment.

Transform Update/Error Minimization

Input: 4 adjacent images:$M1, M2, M3, M4$, and table of pairwise transformations and associated error.

Output: $T^*_{2,3}$, the transformation matrix to align $(M2, M3)$ that minimizes error function $e^*_{2,3}$

Let $T_{2,3}$ be the transformation matrix to align $M2, M3$, and $e_{2,3}$ be the associated error.

1) Compute $T_{1,3}$ and $T_{2,4}$.

2) We wish to find $T^*_{2,3}$ that minimizes $e^*_{2,3}$ by using $T_{1,3}, T_{2,4}$.

3) Let $T^*_{2,3} = [T^{-1}_{1,3} * T_{1,2}]x_1 + [T_{2,4} * T^{-1}_{3,4}]x_2 + [T_{2,3}]x_3 + [I]x_4$

Note that $T^{-1}_{1,3} * T_{1,2}$ and $T_{2,4} * T^{-1}_{3,4}$ are two other estimates of $T_{2,3}$.

Let $e_{1,2,3}$ be the error from aligning with $T^{-1}_{1,3} * T_{1,2}$, $e_{2,3,4}$ be the error from aligning with $T_{2,4} * T^{-1}_{3,4}$.

Let e_i be the error from not aligning (using identity matrix I)

4) Pose the following equation and constraints:

$$\begin{aligned} \text{minimize: } & z = e_{1,2,3}x_1 + e_{2,3,4}x_2 + e_{2,3}x_3 + e_i x_4 \\ \text{constraints: } & x_1 + x_2 + x_3 + x_4 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

5) Solve to find x_1, x_2, x_3, x_4.

6) **return** $T^*_{2,3}$ by plugging x_1, x_2, x_3, x_4 into $[T^{-1}_{1,3} * T_{1,2}]x_1 + [T_{2,4} * T^{-1}_{3,4}]x_2 + [T_{2,3}]x_3 + [I]x_4$