Alignment of Electron Microscope Sections

Roger Zou and Julia Ni

July 17, 2014

1 Overview

Scalable 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 a cube of EM image slices of the brain. See the README for details about the specific 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.

1.1 Computational Complexity

Time complexity: $O(nlogn)$, where n is the number of voxels due to the cross-correlation procedures with the Fast Fourier Transform (FFT). The size of n can be greatly reduced by picking a lower resolution image (directly from the API or by downsampling). Thus cross-correlation is the bottleneck of the alignment pipeline.

2 Pairwise Alignment

Before an entire 3D image cube can be globally aligned, we must compute alignment between pairs of images.
The procedure uses 2D cross-correlation to identify the degree of rotation and magnitude of translations necessary to a pair of images. See section 2.1 for specifics.
One important step in the cross-correlation process is identifying peaks in the cross-correlation that correspond to the translations necessary to align the two images. Difficulties arise when the 'correct' peak is NOT the maximum value; simply max picking would result in completely inaccurate parameters. Our solution is to use a trained support vector machine (SVM) classifier to determine the correct peak among potential candidate peaks. See section 2.2 for specifics.

2.1 Alignment Procedure with Correlation

Iterate through image stack, and apply the procedures described below to each adjacent pair of images.

1. Apply median filtering and histogram equalization to both images.
2. Apply hamming window to both images.
3. Take the Discrete Fourier Transform (DFT), apply a high-pass filter, and then resample the images in log-polar coordinates.
4. Correlate the two images to find the best rho (scaling) and theta (rotation angle) parameters by max picking.

5. Rotate the image appropriately, then correlate the two (rotated) images to find the best translation transformation parameters.
6. Use an SVM to identify the peak in the cross-correlation of the image pair 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 (from 1024x1024 original). In practice, this method greatly improves the running time.

2.2 Peak Identification in the cross-correlation of two images

2.2.1 SVM Training

Using a stack of aligned images, do the following procedure:

1. Randomly sample from each image pair a cropped image with specified minimum size.
2. Generate image from the cross-correlation of the image pair; the image with its associated max value is labeled in the true (aligned) class.
3. Apply a random noticeable rotation and translation to one of the images. Generate image from the cross-correlation of the image pair (now rotated); this image with its associated max value is labeled in the false (unaligned) class.
4. repeat 1,2 as long as needed.
5. Generate a 1x5 feature vector for each test sample. Use these feature vectors to train SVM with a linear kernel.

Once the SVM is trained, it is saved; In the alignment pipeline, the default config settings automatically load this trained SVM for peak classification.

2.2.2 Peak Identification with SVM

1. Divide image I, the cross correlation of two images, into 9 equal parts $I_1...I_9$.
2. Find the coordinate of maximum value in each part, $(x_1, y_1, I(x_1, y_1))...(x_9, y_9, I(x_9, y_9))$, where $I(x, y)$ is the intensity value at coordinate x, y.
3. Sort the coordinates by order of decreasing intensity.
4. Classify I with point (x_i, y_i) for $i = 1...9$ until a true classification. The first true classification is determined to be the actual peak with specific translation parameters. If no true classification are obtained, then the image is deemed to be unaligned.

The described method works the bets compared to other methods, briefly described below.

Other Attempted Methods

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 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?

3.1 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.

3.2 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.

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

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.

4 Attempted, but ultimately unused methods

4.1 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.

4.2 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.

4.3 Error metric with superpixels and Earth Mover's Distance