Alignment of EM sections

Roger Zou, Julia Ni

July 7, 2014

1 Image Enhancement

1.1 Fold Removal

Haven't focused much on this. Right now uses RANSAC to detect linear folds. Images on either side of the fold are split into two images. The bottom right image are two images overlaid (pink, green) after using RANSAC to detect the fold in the top left image.
Problems: Only detects linear folds right now that extend across image.
Improvements: Are non-linear folds/other deformities a priority? Also, there are sometimes patches of very bright marks and patches of very dark marks. This may disrupt alignment with correlation, though unsure by how much. Would impainting help?

2 Global Pairwise Alignment

Computes the pairwise alignment between adjacent two images in an image cube. Optionally first uses feature matching to determine the rotation parameter between two images. Then uses 2D Cross-Correlation to refine the rotation parameter and determine the translation parameters.

2.1 Feature Matching

1. gaussian blur, then downsample (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 rotation to image for input to 2D cross-correlation step.

Problems: effectiveness decreases for images that are more dissimilar. When images have noise and blemishes, it struggles to identify any features at all, or the matches are spurious. Not using this step doesn't seem to make much of a difference right now.
Improvements: Are there ways of using entire cell structure features (such as entire mitochondria and other organelles) to use 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.

2.2 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 DFT, then high-pass filter, then resample in log-polar coordinates
3. compute phase correlation to find best rho (scaling) and theta (rotation angle)

4. refine initial rotation parameter (computed from feature matching previously) by using the theta value from correlation

5. rotate the image appropriately, then compute normalized cross correlation to find translation parameters

6. use support vector machine to identify peak in the cross correlation matrix of images that accurately correspond to the actual translation parameters

7. save pairwise affine transformation matrix

The time complexity is $O(nlogn)$, where n is the the number of voxels. The upper bound is because of the Fast Fourier Transform 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. Scaling by 0.5 effectively decreases the constant in front of $nlogn$ in the running time by 4.

Problems: For images that are more dissimilar, identifying the rotation parameter is a challenge; The rotation identified is often close to the true rotation, but not close enough (see bottom). For most image inputs, it appears that rotations are small to begin with, so this usually isn't a problem. However, the problem sometimes occurs when I artificially add large rotations to image inputs before alignment.
Improvements: Are there better ways to identify the rotation angle needed to align images? Are there ways to improve the correlation strength? Some humans (or at least I) match outlines of noticeable cell structures. High-pass filtering ideally helps, but the noise and blemishes would probably pass through as well. One idea: selectively darken regions that we don't want to matter as much in the correlation?

2.2.1 Peak Detection

Uses a SVM to classify a local maxima as a peak or non peak. Simply picking the maximum values will not always work, 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, and performs the best compared to other methods. Attempted Methods to find peaks in the cross correlation of two images:

1. Identifying maximum values. However, this won't always work, as the example on the right demonstrates.

2. Correlating the cross correlation of two images with a normal distribution. This is attempting to factor in shape. In theory this should accentuate the true peaks, but this won't always work 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. Support Vector Machine trained on 5 features: number of pixels in image, area of binary thresholded region, max gradient, max laplacian, and skewness of the histogram. The cross correlation images is divided into 9 subsections. The maximum is identified in each, and evaluated with the SVM. The maximum value predicted by the SVM to be a peak is labelled as the actual peak. If no maximum values are predicted by the SVM to be a peak, then no peak exists. This appears to work the best. It accurately identifies peaks even if not the global maximum and the running time is almost instantaneous.

2.3 Global Stack Alignment

Uses the pairwise transformation parameters to compute global transformation parameters. Starts with the first image and builds the aligned stack iteratively.
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?

2.3.1 Tune Transformation Parameters

This attempts to improve the estimate of the rotation amount by minimizing an error function (usually mean squared error). This is also important because log-polar sampling to determine the best rotation parameter is discretized, which means the best rotation angle may not be available at the cross correlation step. Thus, this step iterates over a small range of rotation and translation parameters surrounding the original estimated parameters. It then picks the rotation angle and translation amount that minimizes the error function.

2.3.2 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 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. The transformation parameter that minimizes an 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.