Automatic Assembly of TEM Mosaics and Mosaic Stacks Using Phase Correlation

Pavel A. Koshevoy, Tolga Tasdizen, Ross T. Whitaker

UUSCI-2007-004

Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA

April 20, 2007

Abstract:

This paper discusses automatic Transmission Electron Microscopy (TEM) image registration, TEM slice assembly via tile mosaicking, and TEM volume assembly via slice to slice registration. Several algo- rithms are presented, including an algorithm for mosaic layout of an unordered set of tiles, an algorithm for distortion correction, and an image processing algorithm for a coarse edge and blob detection.

Automatic assembly of TEM mosaics and mosaic stacks using phase correlation

Pavel A. Koshevoy, Tolga Tasdizen, Ross T. Whitaker

April 19, 2007

Abstract

This paper discusses automatic Transmission Electron Microscopy (TEM) image registration, TEM slice assembly via tile mosaicking, and TEM volume assembly via slice to slice registration. Several algorithms are presented, including an algorithm for mosaic layout of an unordered set of tiles, an algorithm for distortion correction, and an image processing algorithm for a coarse edge and blob detection.

1 Motivation

Transmission Electron Microscopy (TEM) brings several challenges to automatic image registration.

An electron microscope rarely has a large enough field of view to cover the region of interest with reasonable detail. Therefore, the region of interest has to be imaged as a sequence of tiles, following some overlapping tile pattern. The imaging process introduces distortion into each tile. Unfortunately, the distortion is typically not the same from tile to tile, therefore each tile has to be unwarped individually.

A bigger issue with the slice to slice registration arises from the fact that each slice actually represents a different cross section of tissue, therefore adjacent slices are not expected to match exactly. Additionally, each slice may undergo a different distortion during cutting, and some slices may be destroyed during cutting. To make matters worse, tile distortion correction during the slice mosaic refinement introduces artificial warping into each slice. Slices are arbitrarily oriented when they are put under the electron microscope, which means that slice to slice registration has to find the correct orientation, translation and distortion correction parameters between arbitrarily oriented warped images of different tissue slices.

Also, as the tissue slices have to be stained with a contrast agent, the images often have different contrast from slice to slice, and in some cases the contrast may be so poor that a traditional contrast enhancement algorithm CLAHE would introduce artifacts into the image making the slice to slice registration impossible.

Dealing with any of the above issues manually is a daunting task.

2 Problem statement

Given a large number of tiles specified in no particular order, a slice mosaic must be constructed and individual tiles must be corrected for distortion. This is the global problem that can be split up into slightly more manageable sub-problems:

- Find pairs of matching tiles.
- Build a rough estimate of the mosaic without distortion correction.
- Refine the mosaic by unwarping all tiles simultaneously.

Slice to slice registration presents a slightly different set of sub-problems:

- Find a rough estimate of the slice to slice registration.
- Keep one section fixed and unwarp the other section.

Once all the slice to slice pairs are registered, they must be stacked into a volume by cascading the slice to slice transforms.

3 Description of the mathematics and algorithms

3.1 Matching pairs of tiles

Finding matching tiles amounts to finding tiles with highest cross-correlation. The method for finding matching tiles implemented in this application is based on a Phase Correlation technique described by Girod and Kuo[1]. The technique is very straight forward, but it has an important prerequisite - it requires that the width and height of the two tiles must match. If that is not the case, one or both of the tiles must be padded on the bottom and on the right side with zeros until both of the tiles have matching dimensions as follows: given unpadded tiles U_0 and U_1, padded tiles S_0 and S_1 are generated such that $width\left(S_0\right) = width\left(S_1\right) = \max\left(width\left(U_0\right), width\left(U_1\right)\right)$ and $height\left(S_0\right) = height\left(S_1\right) = \max\left(height\left(U_0\right), height\left(U_1\right)\right)$.

Having satisfied the prerequisite by padding the tiles, the tiles are transformed into the frequency domain by Discrete Fourier Transform $F_0 = F\left\{S_o\right\}$ and $F_1 = F\left\{S_1\right\}$. The Discrete Fourier Transform functionality is provided by the FFTW[4] library. Once the tiles have been transformed, the cross power spectrum between S_1 and S_0 is calculated as

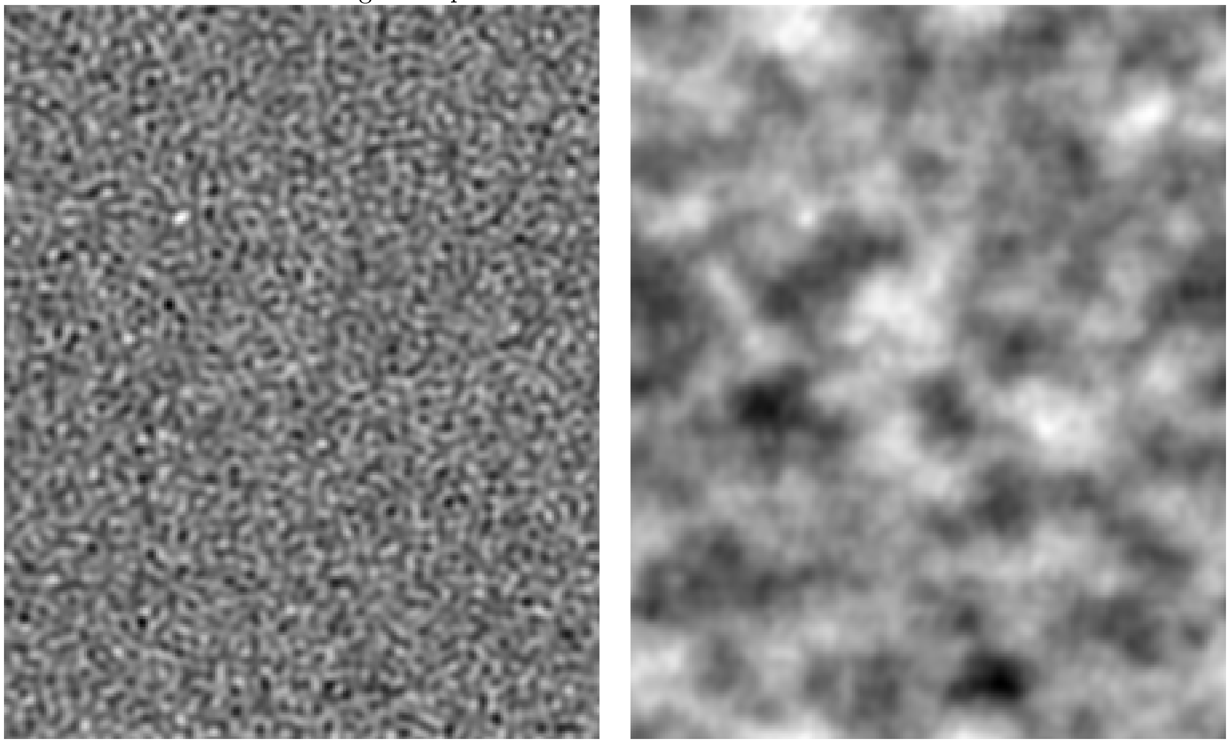
$$\Phi_{10} = F_1 \times F_0^*$$

where F_0^* is the complex conjugate of F_0. The the cross power spectrum is normalized as follow

$$P = \frac{\Phi_{10}}{\sqrt{\Phi_{10} \times \Phi_{10}^*} + \epsilon}$$

where ϵ is a small number greater than zero added to avoid division by zero. The Girod and Kuo paper addresses a slightly different problem than the one targeted by our application. The technique described in the paper is intended for tracking a moving object. One of the difficulties of the tracking problem is that the background behind the object changes. The mosaicking problem typically does not suffer from this obstacle. During early experimentation we attempted to use the non-normalized cross power spectrum directly as $P = \Phi_{10}$. This was found to be unacceptable because the peaks in the cross correlation image are poorly defined. The comparison of the phase correlation and cross correlation can be seen in figure 1.

Figure 1: phase correlation vs. cross correlation



phase correlation

cross correlation

The phase correlation is the inverse Fourier transform of the normalized cross power spectrum.

$$PDF\,(x,y) = \Re\left(F^{-1}\left\{P\right\}\right)$$

The phase correlation corresponds to the probability density function (PDF) that tile S_1 matches with tile S_0 displaced by vector $[x\,y]^T$. We will refer to this function as the displacement PDF. Thus, in order to find the displacement vector it is necessary to find the coordinates $[x_{max}\,y_{max}]^T$ of the global maximum of this function.

Finding the maximum of the displacement PDF is non-trivial. This is due to the fact that for most electron microscopy images the PDF is very noisy. Also, the PDF may contain several peaks of comparable magnitude. This may happen for mismatched images as well as for matching images due to the repetitive texture of the microscopy images. The technique described in the Girod and Kuo paper mentions a simple thresholding method used to suppress the negative and insignificantly small values of the PDF. The method currently implemented in the mosaicking application is similar, but has several important features that are worth pointing out.

Early experimentation with the PDF has shown that identifying the maxima becomes significantly easier after blurring the PDF to remove the high-frequency noise. The blurring is carried out in the frequency domain, where it corresponds to a multiplication by a low-pass filter

$$PDF\,(x,y) = \Re\left(F^{-1}\left\{P \times Filter\,(r,s)\right\}\right)$$

where $r \in \left[0, \sqrt{2}\right]$ and $s \in [0, r]$. When $s = 0$ the filter behaves exactly like the ideal low-pass filter, passing unaffected frequencies in the range $[0, r]$ and attenuating completely frequencies in the range (r, ∞). When $s > 0$ the filter passes frequencies in the range $[0, r - s]$ completely unaffected, frequencies in the range $(r + s, \infty)$ are completely attenuated, and frequencies in the range $(r - s, r + s]$ are attenuated according to the function

$$attenuation\,(f) = \frac{1 + \cos\left(\pi \frac{f-(r-s)}{2s}\right)}{2}$$

which provides a smooth transition from zero attenuation at $f = r - s$ to full attenuation at $f = r + s$. This low-pass filter results in zero total power loss in the frequency range $[0, r]$, because the attenuation incurred in range $[r - s, r]$ is canceled out by the power leakage from range $[r, r + s]$ due to aliasing.

More experimentation has shown that blurring the tiles prior to calculating their corresponding PDF reduces the number of false maxima in the PDF. The tiles are blurred in the frequency domain as follows

$$\begin{aligned} F_0 &= F\left\{S_0\right\} \times Filter\,(r,s) \\ F_1 &= F\left\{S_1\right\} \times Filter\,(r,s) \end{aligned}$$

and the rest of the calculations are carried out as described above. The parameters r and s used for blurring the tiles and the PDF can be tuned. In the current implementation the values $r = 0.5$ and $s = 0.1$ are used for the tiles, and $r = 0.4$ and $s = 0.1$ for the PDF.

Having blurred the PDF, it is necessary to select a good threshold value in order to isolate a set of pixels corresponding to the global PDF maximum. We assume that the number of pixels belonging to the maximum is approximately 1% of the total number of PDF pixels, but it may not be less than 5 pixels or greater than 64 pixels. The lower bound restriction is imposed in order to avoid thresholding values where only one maximum pixel is left. One pixel does not carry enough information about the rest of the structure of the PDF. When 5 pixels are grouped together, it is fairly obvious that there is only one strong maximum in the PDF. If the pixels are scattered across the PDF, it is likely the PDF does not have a well defined maximum. The lower bound on the number of pixels belonging to the PDF maximum is necessary in order to deliver the information regarding the distribution of these pixels within the PDF. One or two pixels do not carry enough information. The upper bound on the number of pixels applies to larger images. If too many pixels are allocated to the PDF maxima, the computational burden involved in the classification of the clusters increases. The upper limit of 64 pixels guarantees that no PDF could ever contain more than 64 maxima. Thus

$$pixels_{maxima} = \min\left(64, \max\left(5, \frac{area\,(PDF)}{100}\right)\right)$$

where $area\,(PDF)$ corresponds to the total number of pixels in the PDF image.

To find the threshold value that would provide this number of pixels, it is necessary to build a cumulative histogram of the PDF pixel values. The current implementation uses 1024 histogram bins. Although the importance of this parameter has not been explored in the context of our application, we can assume that more bins will give us a more accurate estimate of the threshold value. The cumulative histogram is searched for the bin containing at least

$$area\,(PDF) - pixels_{maxima}$$

number of pixels. The minimum pixel value associated with that bin is the optimal threshold value that we need.

Once the PDF is thresholded, a small fraction of the pixels belonging to the maxima are isolated into one or more clusters. Next, pixels are classified into clusters based on an 8-connected neighborhood stencil. Once all of the clusters have been identified, the clusters that are broken up across the PDF boundary are merged together. This step is required because the Discrete Fourier Transform assumes that the signal is periodic; therefore, the PDF is also periodic. After all of the pixel clusters are identified, the coordinates of the PDF maxima are calculated as the centers of mass of the corresponding clusters. The value of each maximum is calculated as the total mass of the cluster divided by the number of pixels in that cluster. This process results in a list of several maxima with varying coordinates and values. The list is sorted in descending order, so that the highest maximum is at the head of the list.

Given a list of maxima points present in a particular PDF, a simple heuristic is applied to decide whether the tiles that produced this PDF in fact match. Matching tiles would ideally produce only one maximum. However, due to the inaccuracy in the selection of the thresholding value, it is very likely that there will be several maxima. This is also the case when the tiles being matched have undergone a distortion. During experimentation an important observation was made that mismatching tiles produce a PDF with several maxima points at roughly the same value, while the PDF of two matching tiles produces one maximum significantly higher than the rest. This result suggests a very simple algorithm to decide whether the PDF corresponds to two matching tiles. The dissimilarity of the PDF maxima with respect to the best PDF maximum is calculated as

$$dissimilarity = \frac{\max_{best}(PDF)}{\max_i(PDF)} - 1$$

The *dissimilarity* of two perfectly similar maxima is equal to 0. Whenever *dissimilarity* exceeds a given threshold the corresponding maximum is removed from the list. In current implementation, the *dissimilarity* threshold is set to 1; thus, maxima which are more than 2 times smaller than the highest maxima in the list are discarded. If the list contains only one maximum, we assume that the tiles match and proceed to calculate the corresponding displacement vector. If there is more than one maximum left in the list after this filtering, it is very likely that the tiles do not match, or one of the tiles is self-similar and may match the other tile in several places. Due to distortion, it is possible that no matching tiles will be found with exactly one maximum. In that case the match with the fewest number of maxima is considered. Significantly distorted tiles typically have 2 to 4 valid maxima corresponding to small shifts from the true displacement vector. The current implementation of the mosaicking application considers at most 3 maxima per match.

In order to find the displacement vector, it is not enough to simply find the maximum of the displacement PDF. The coordinates $[x_{max}\, y_{max}]^T$ are always positive, yet the displacement vector may very well have negative components. As mentioned earlier, the Discrete Fourier Transform assumes that the signal is periodic, therefore the cross-correlation between the tiles corresponds to cross-correlation of two periodic tiles. Once the coordinates of the maximum $[x_{max}\, y_{max}]^T$ are known, there are four possible permutations of the displacement vector that could produce the corresponding high cross-correlation between the tiles. The permutations are

$$\begin{aligned} T_{00} &= \begin{bmatrix} x_{max} \\ y_{max} \end{bmatrix} \\ T_{10} &= \begin{bmatrix} x_{max} - width\,(S_0) \\ y_{max} \end{bmatrix} \\ T_{01} &= \begin{bmatrix} x_{max} \\ y_{max} - height\,(S_0) \end{bmatrix} \end{aligned}$$

$$T_{11} = \begin{bmatrix} x_{max} - width\left(S_0\right) \\ y_{max} - height\left(S_0\right) \end{bmatrix}$$

The current implementation of the application chooses the best permutation based on the normalized cross correlation image metric. The best permutation corresponds to the lowest metric value (the least mismatch between the tiles). The metric is evaluated against unpadded tiles U_0 and U_1, yet the displacement permutations are based on the dimensions of the padded tiles S_0 and S_1, which means that some of the permutations may not overlap the unpadded tiles at all. In consequence, permutations can be discarded early based on the amount of overlap between the tiles. The amount of overlap is computed as the ratio of the area of the overlap region to the area of the smaller of the two tiles. Thus, when one tile overlaps another entirely, the overlap is equal to 1. Displacement vectors resulting in less than 5% of overlap are discarded without further consideration. This decision is based on the fact that typical tiles will have 20% to 30% of overlap along the edges of the tile, and approximately 10% to 5% of overlap at the corners.

3.2 Initial mosaic layout

Prior to deducing the tile layout it is necessary to find pairs of matching tiles. The runtime complexity of the current algorithm for finding the matching tiles is $O\left(n^2\right)$. The performance of this algorithm may be improved, but not without sacrificing some robustness in finding the correct tile matches and rejecting the mismatches. Why this is the case will become more clear after the current algorithm is explained in greater detail.

The algorithm tries to find the best possible mapping from the image space of one tile into any other tile. This is accomplished by cascading the mappings via intermediate tiles. For example, there may exist a mapping $U_0 : U_1$ between tiles U_0 and U_1, and another mapping $U_1 : U_4$ between tiles U_1 and U_4. A mapping $U_0 : U_1 : U_4$ between tiles U_0 and U_4 can be created via the intermediate tile U_1. The number of intermediate steps in a mapping from one tile to another will be referred to as the cascade length from now on. Given n tiles, there may be at most $n-2$ intermediate steps in a mapping between any 2 tiles. Of course, this is only the upper bound on the cascade length. There are no guarantees that a mapping with a given cascade length exists between any 2 tiles. However, the fact that there may be redundant mappings between any 2 tiles presents a great opportunity to select the best mapping possible.

The algorithm proceeds as follows. First, pairs of matching tiles are found. Finding just one match for every tile is not enough, because that does not provide any redundant mappings between the tiles. This is the reason why the algorithm has $O\left(n^2\right)$ run time complexity. One way to speed up the algorithm is to limit the number of redundant mappings to some fixed maximum number per tile. Allowing a maximum of just 2 mappings per tile may introduce enough redundancy to correct for mismatches while also speeding up the matching process.

The mappings between the tiles are stored as connections in a graph of tiles. Each mapping (connection) is weighed according to the normalized cross correlation image metric. Next, redundant mappings with cascade length 1 to $n-2$ are found. There may be more than one such mapping, therefore it is useful if the process is explained with an example. Assume there exists a function

$$C\left(U_i : U_j\right) = cost$$

that evaluates the cost of a mapping between tiles U_i and U_j. Given the following sample mappings

$$\begin{aligned} C\left(U_0 : U_1\right) &= 278 \\ C\left(U_0 : U_2\right) &= 311 \\ C\left(U_1 : U_4\right) &= 160 \\ C\left(U_2 : U_4\right) &= 121 \\ C\left(U_0 : U_4\right) &= 3419 \end{aligned}$$

it is most likely that the mapping $U_0 : U_4$ is mismatched. There are 2 possible alternative mapping from tile U_0 to U_4. The cost is set to the maximum cost of the intermediate mapping costs. In the context of this example, this means that

$$C\left(U_0 : U_1 : U_4\right) = \max\left(C\left(U_0 : U_1\right), C\left(U_1 : U_4\right)\right) = 278$$

$$C(U_0 : U_2 : U_4) = \max(C(U_0 : U_2), C(U_2 : U_4)) = 311$$

The mapping with the least cost (in this case $U_0 : U_1 : U_4$) is preferred even when it has greater cascade length.

In order to generate the mosaic, it is necessary to select the target tile into which every other tile will be mapped. This is done by considering the total cost of the target tile candidates. The total cost is calculated as the cumulative cost of the mapping from the target tile to every other tile in the mosaic. The candidate with the lowest total cost becomes the target tile.

3.3 Distortion correction

In order to correct for distortion each tile has to be unwarped.

During the earlier stages of the development, several continuous polynomial transforms were explored, in particular a bi-variate cubic Radial Distortion transform and a bi-variate cubic Legendre polynomial transform. These transforms suffer from a trade-off where the stability of the transform is related inversely to the degree of the polynomial. Higher degree polynomial transforms may explode even when they are properly normalized and centered, while lower degree polynomial transforms limit the amount of distortion correction that may be achieved. Bi-variate polynomial transforms of degree greater than one may not have an analytic inverse, therefore an iterative numeric inverse calculation must be used[2]. Simultaneous numerical optimization of several polynomial transforms is computationally expensive.

Our latest approach uses a discontinuous transform. Each tile is sampled onto a coarse uniform triangle mesh. Each vertex in the mesh stores two sets of coordinates – the local tile coordinates and the mosaic space coordinates. The image is warped by changing the mosaic space coordinates directly. Anyone familiar with texture mapping in OpenGL will readily recognize the similarity here. The tile space coordinates correspond to the OpenGL texture coordinates, and the mosaic space coordinates correspond to the OpenGL triangle vertex coordinates.

To map a coordinate from the mosaic space into the tile space, the tile mesh is searched for the triangle containing the given mosaic space point. This is similar to ray/triangle intersection operation carried out in Raytracing. The barycentric coordinates of the intersection point are used to calculate the corresponding tile space point by interpolating the tile space vertex coordinates. Acceleration datastructures commonly used in Raytracing are also applicable here. The current implementation uses a trivial 2D grid acceleration datastructure.

The mapping from tile space into mosaic space is trivial due to the uniform structure of the triangle mesh in the tile space. One has to find the mesh quad containing the tile space point and perform a bi-linear interpolation between the mosaic space coordinates of the quad vertices.

At each vertex in the mesh, a small image neighborhood of the tile is sampled in the mosaic space. A corresponding neighborhood is sampled from all of the tile neighbors in the mosaic. The neighborhood has to be only as large as necessary to capture a meaningful amount of image texture for phase correlation to work. Currently, we downscale tile images by a factor of 8 (roughly 400×500 pixels) and the use 96×96 pixel neighborhoods. The mesh nodes are spaced at approximately one third of the neighborhood size, so a typical microscopy tile is covered by a 13×16 vertex mesh (420 triangles).

The two neighborhoods are matched as described in section 3.1 on page 2. The displacement vectors produced by this matching are used to correct the mosaic space coordinates of the vertex. One vertex neighborhood may be matched with more than one neighbor tile, which means the displacement vectors for that vertex would have to be combined. Because we are trying to warp all tiles simultaneously, the displacement vectors computed for neighboring tiles will overlap, causing the distortion correction to overshoot. This means that the displacement vectors have to be scaled down according to the number of the overlapping neighbors at that point, $scale = 1/(1 + NumberOfDisplacements)$.

Since it is possible for tile matching to produce mismatches, the displacement vectors calculated at each vertex are filtered using a median filter to remove the outliers. Holes in the displacement vector image are filled in using dilation. The displacement vectors are further denoised with a Gaussian smoothing filter. All this post-processing necessitates several passes of the algorithm to ensure convergence. For our current datasets we've found 2-3 passes to be sufficient. Actually, even a single pass of this algorithm produces better results than our best effort using the traditional ITK style mean pixel variance metric optimization of several Legendre polynomial transforms simultaneously.

3.4 Slice to slice registration

Slice to slice registration is very similar to distortion correction except for two differences. Since the orientation of the slices is arbitrary we cannot use image correlation to estimate the image to image translation parameters. Instead, we perform a brute force search for tile translation/rotation parameters at a very coarse scale by downscaling the slices to tiny thumbnails about 128×128 pixels each.

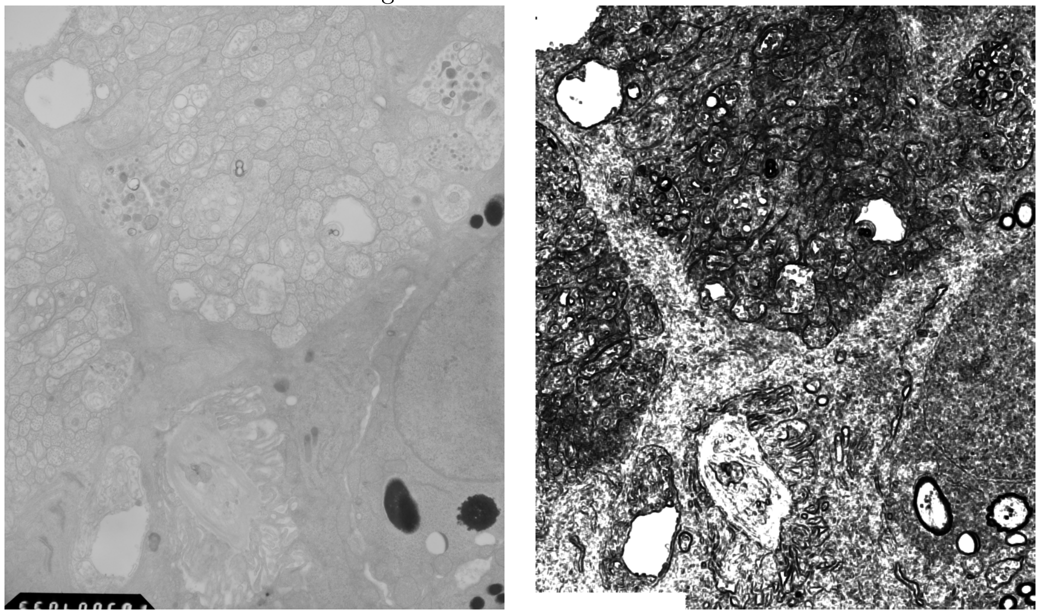
When the slices are downscaled to tiny thumbnails virtually all image texture is lost, even when the images were contrast enhanced. Therefore, the brute force slice to slice registration is carried out on preprocessed images. The blob enhancement algorithm enhances features at coarse scales such that they would not get washed out when downscaling the image, see figure 2, and figure 3 on the next page for illustration.

The blob enhancement algorithm is as follows:

1. The image is partitioned into a regular cell grid of roughly 17×17 pixels per cell.
2. Mean pixel variance is calculated within each cell.
3. The cell variances are sorted and the median variance is selected.
4. The algorithm iterates through all image pixels, and for each pixel calculates mean pixel variance within the local 17×17 pixel neighborhood centered at the pixel. The output pixel value is computed as $min(3, (medianVariance + 1)/(localVariance + 1))$.

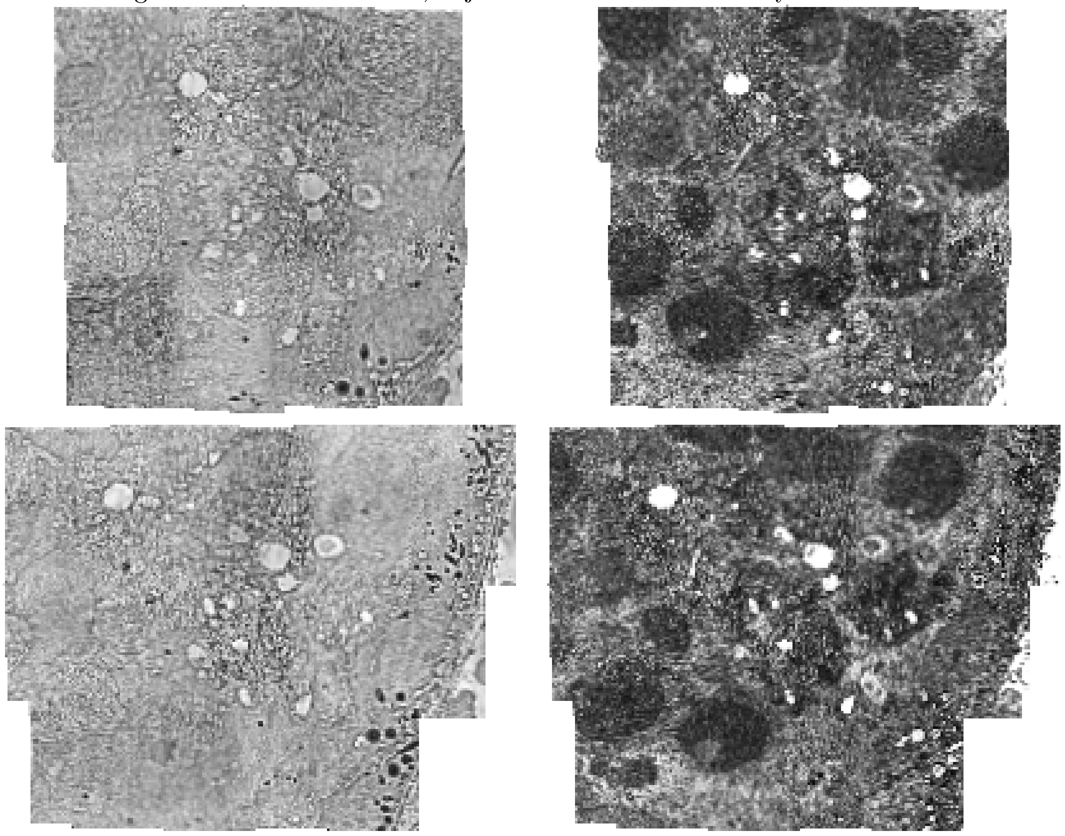
The result of this algorithm is that it enhances regions with greater than median variance – the edges, which become black in the output image. The algorithm also enhances regions with lesser than median variance – the flat spots (blobs) which become white in the output image.

Figure 2: blob enhancement



original tile 48691 blob enhanced tile 48691

Figure 3: CLAHE vs blobs, adjacent slices scaled down by a factor of 64



When scaled down by a factor of 64, the texture in the CLAHE processed slices is washed out, while the coarse texture properties brought out by the blob enhancement algorithm remain visible. The dark blobs correspond to the high variance texture regions, and the white blobs correspond to the flat spots (low variance).

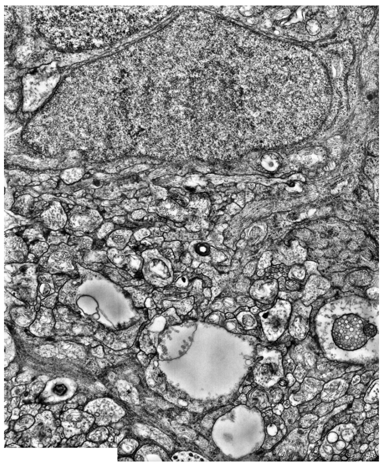
The brute force search is accelerated using the phase correlation to determine slice to slice translation parameters. The moving slice is rotated in increments on 1 degree and matched against the fixed slice as described in section 3.1 on page 2. The quality of the match is evaluated via the normalized cross correlation image metric. The rotation and translation parameters corresponding to the best brute force match metric are used to initialize the mesh transform of the moving slice. The transform is then refined as explained in section 3.3 on page 6, except the displacement vectors are applied to the moving slice only.

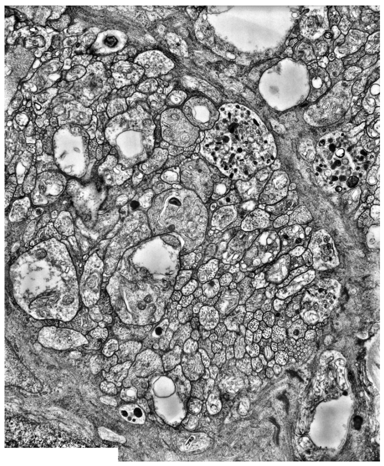
4 Results

4.1 Tile matching

Figure 4 on the following page shows two matching image tiles. Figure 5 on the next page shows the displacement PDF corresponding to these tiles, and highlights PDF maxima. There are a total of 50 maxima isolated in the PDF. Filtering leaves only 2 eligible maxima (highlighted in red and green), which indicates that the tiles match.

Figure 4: matching tiles


tile 48685


tile 48690

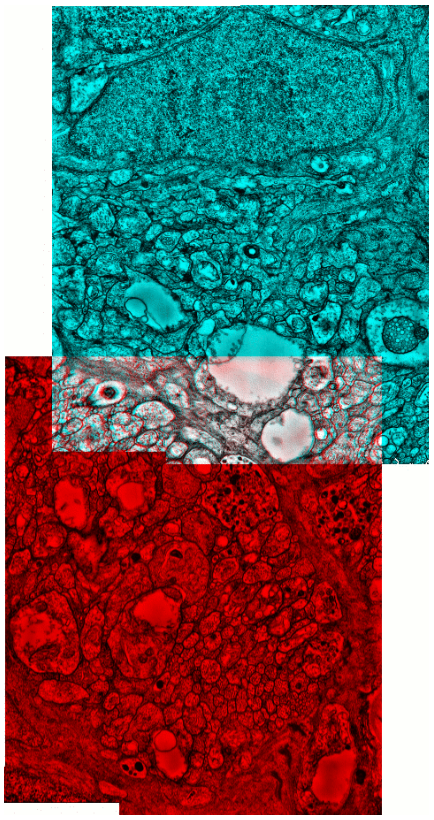
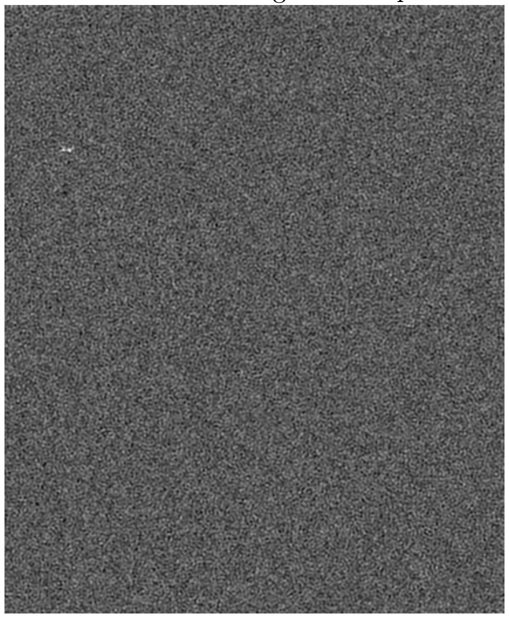
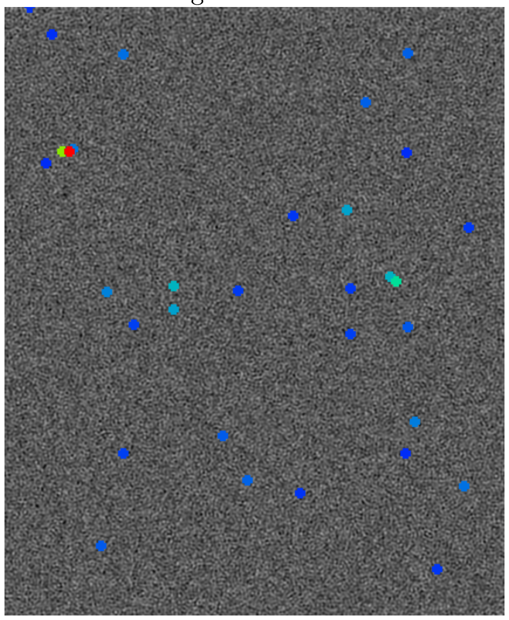

mosaic 48685:48690

Figure 5: displacement PDF for matching tiles

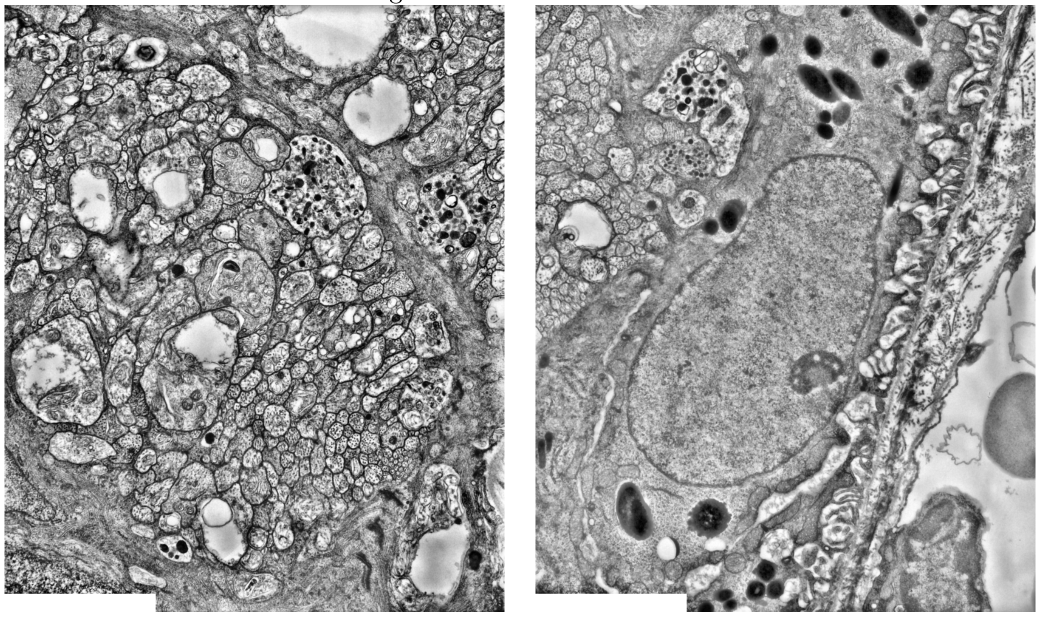

PDF 48685:48690


Colormapped PDF maxima.

The colder colored maxima (blue) are filtered out, leaving only two warm colored (red and green) maxima. One of the remaining maxima corresponds to the translation vector for matching tiles.

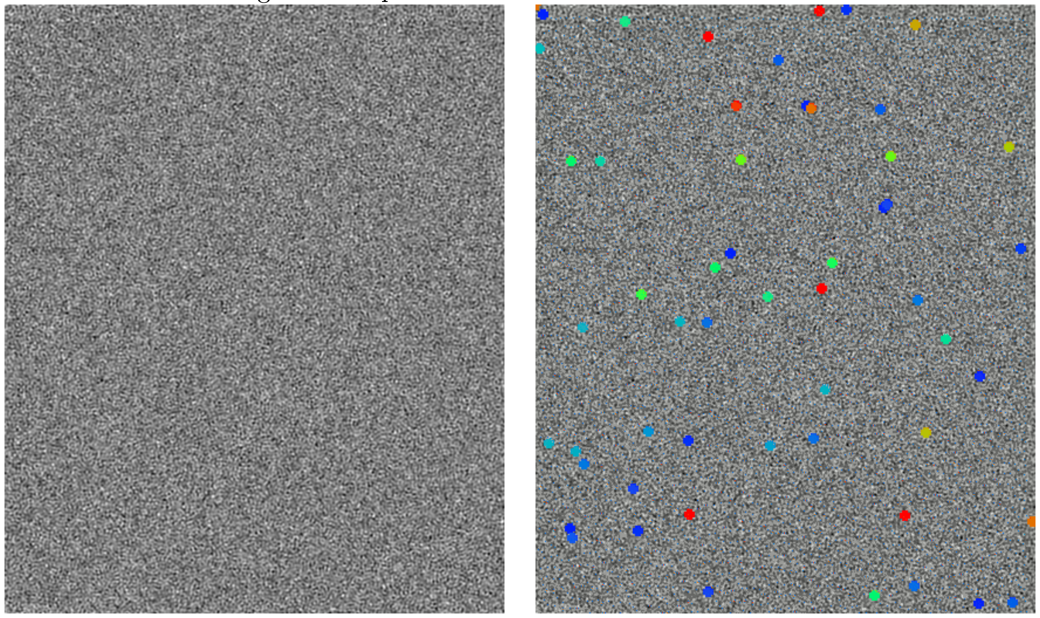
Figure 6 shows two mismatched tiles. Figure 7 shows the corresponding displacement PDF and PDF maxima. There are 59 maxima isolated in this PDF. After filtering there are still 14 maxima left. Ideally there would be less than 4 maxima left, therefore this PDF indicates that the tiles do not match.

Figure 6: mismatched tiles



tile 48690 tile 48692

Figure 7: displacement PDF for mismatched tiles



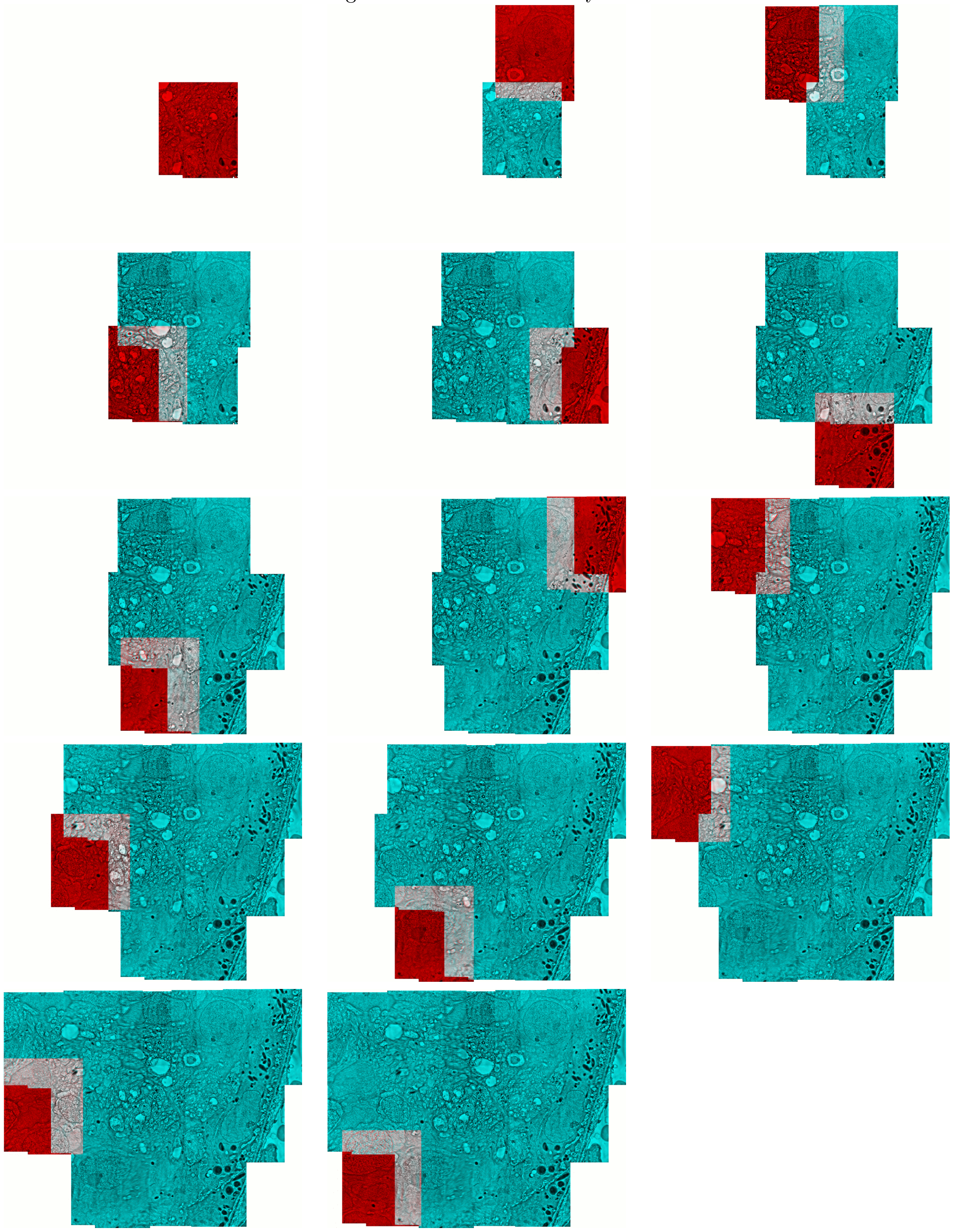
PDF 48690:48692 Color-mapped PDF maxima

There a 14 warm colored PDF maxima left after filtering, indicating that the tiles do no match.

4.2 Tile layout

Figure 8 on the next page illustrates the order in which the tiles are added to the mosaic. The algorithm lays out new tiles (shown in red) such that they have significant overlap with previous tiles (shown in blue). The incremental mosaic layout allows us to refine the mosaic as each tile is added to it, although currently we do not exercise this capability.

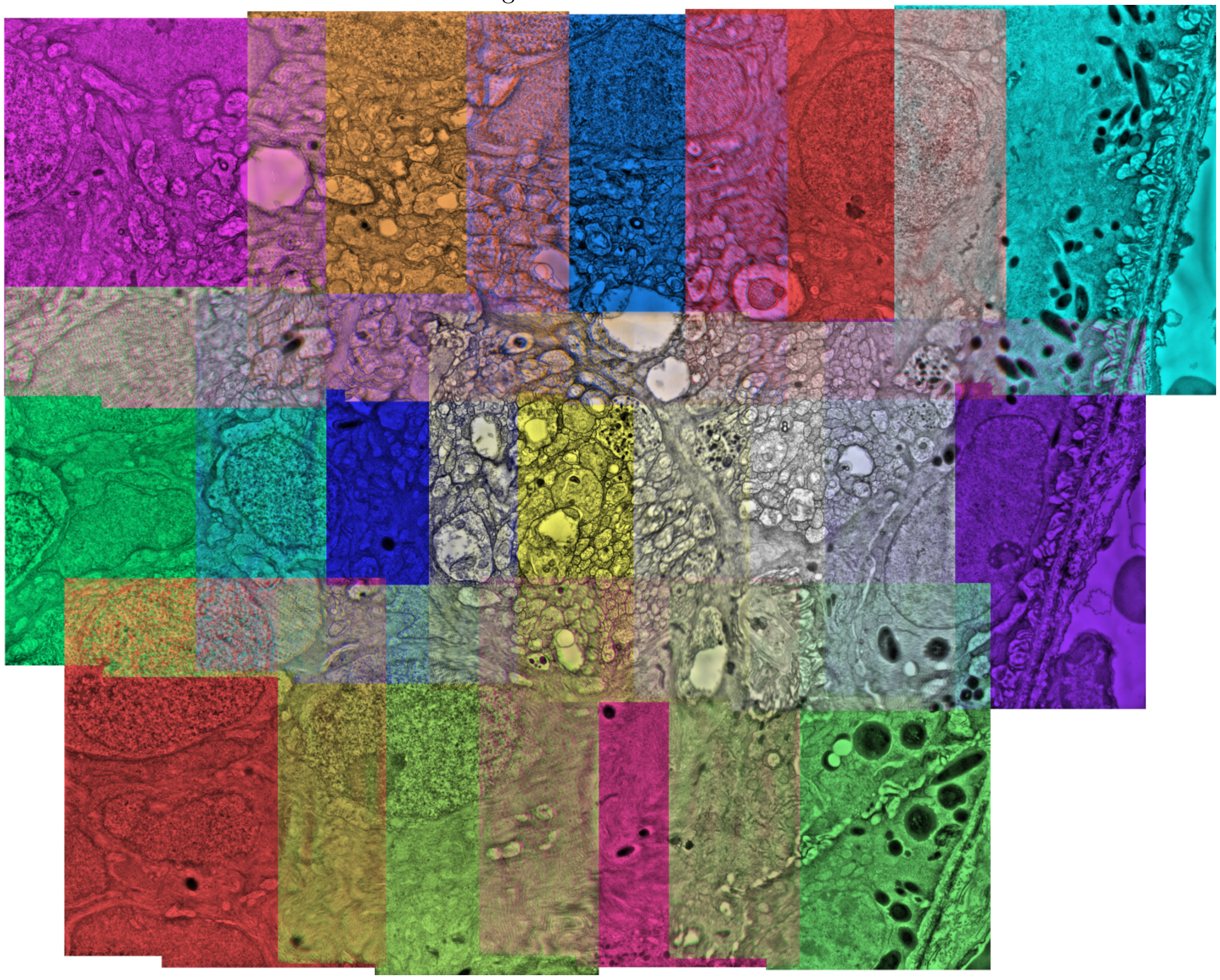
Figure 8: incremental tile layout



4.3 Distortion correction

Figure 9 on the following page illustrates the initial mosaic layout.

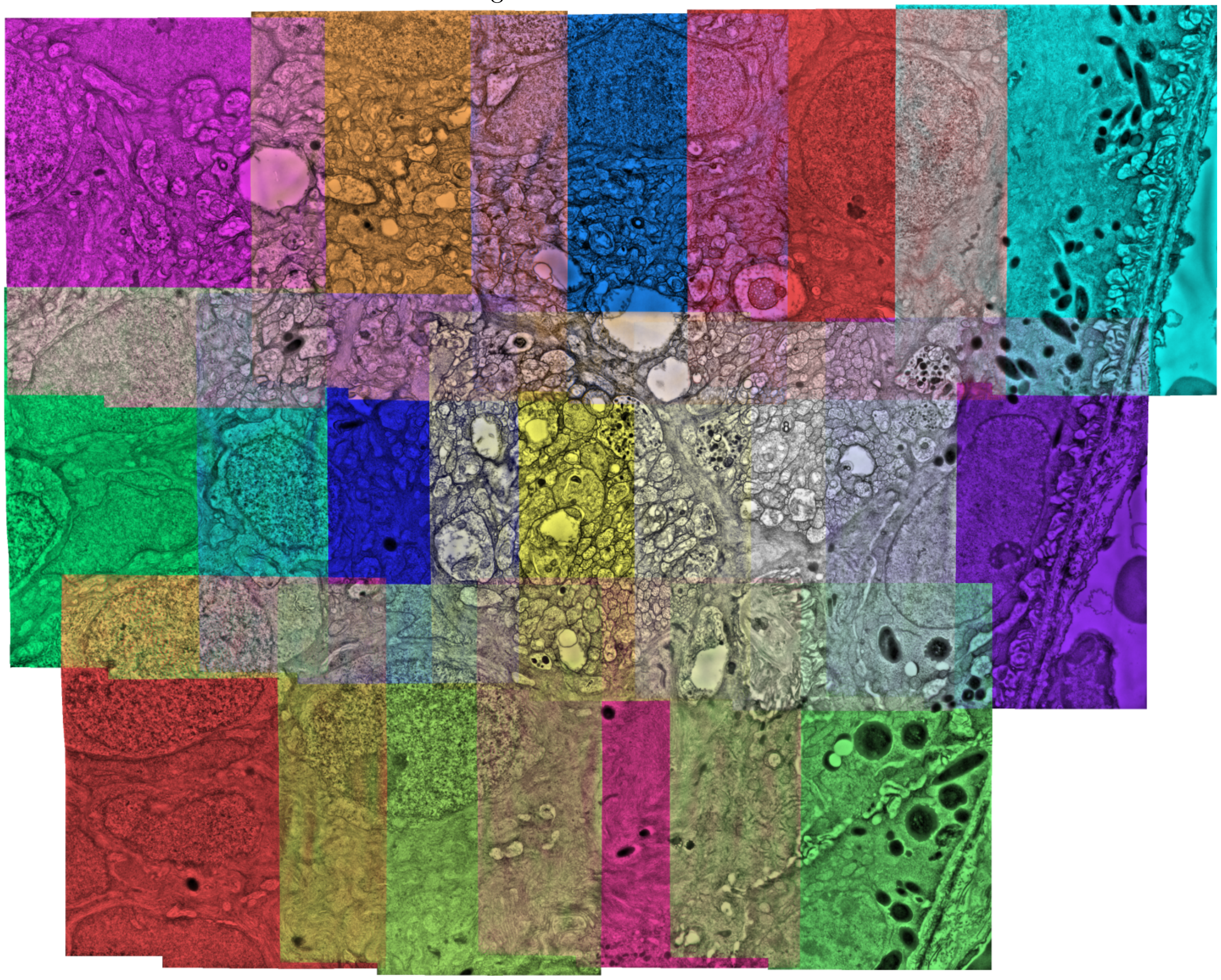
Figure 9: initial mosaic



The initial mosaic, mean pixel variance is 593.

Figure 10 on the next page illustrates the refined mosaic.

Figure 10: refined mosaic

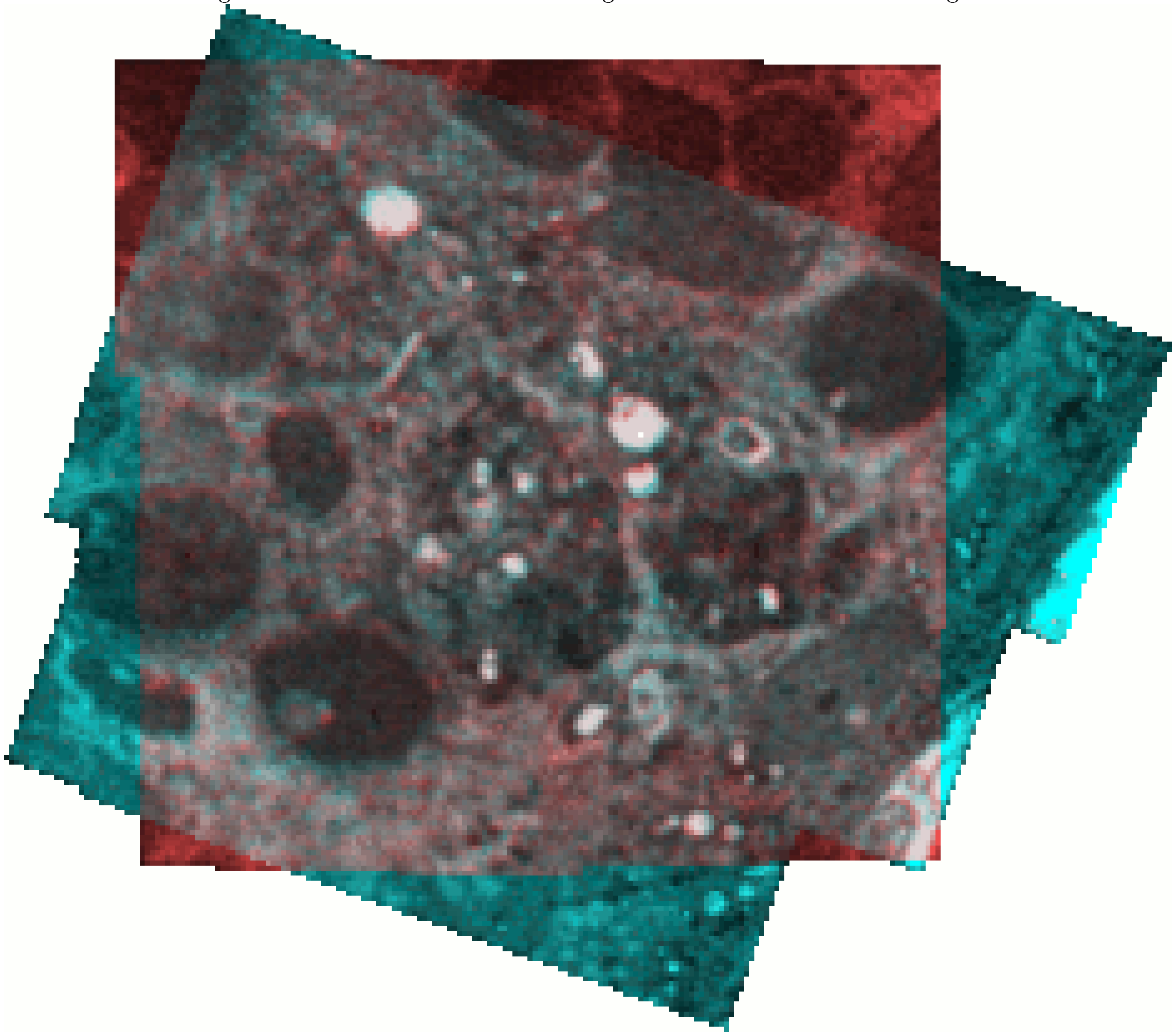


The refined mosaic, mean pixel variance is 213.

4.4 Slice to slice registration

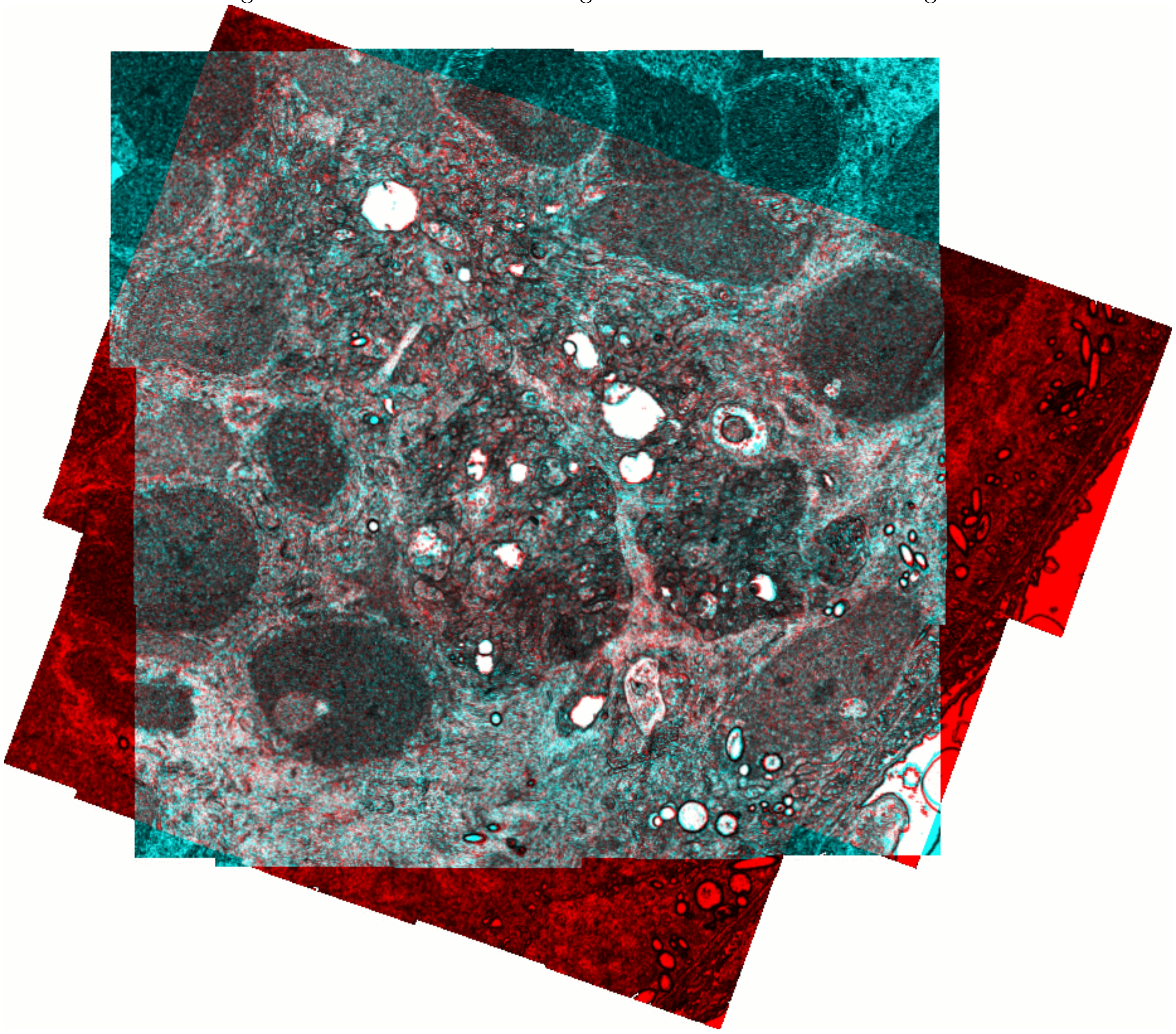
The slices were assembled from the blob enhanced image tiles, and downscaled to about 160×160 pixel thumbnails. The moving slice is rotated in one degree increments, and matched against the fixed slice using the phase correlation as described in section 4. Figure 11 on the following page illustrates the brute force registration results.

Figure 11: brute force slice to slice registration of blob enhanced images



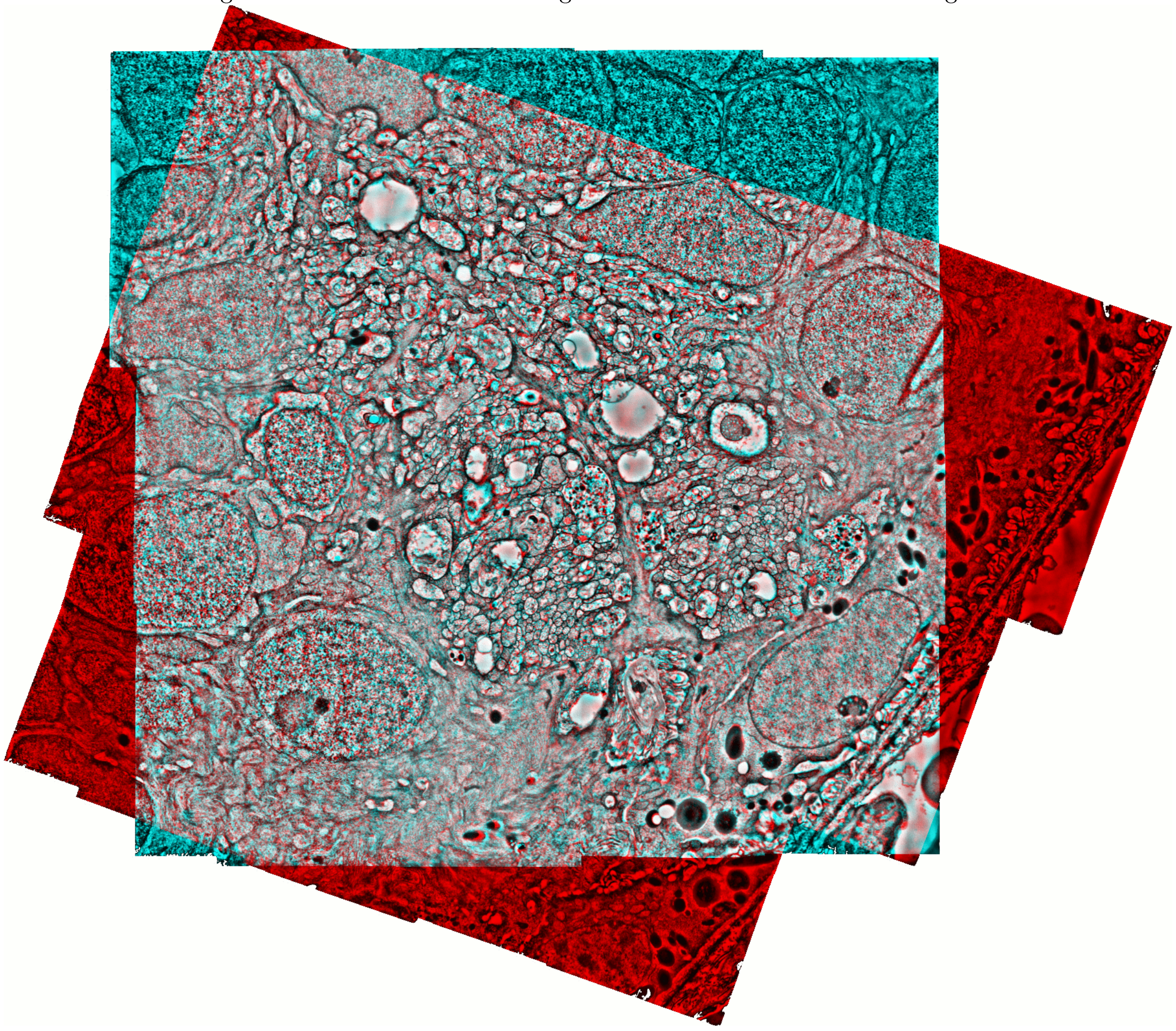
The mesh transform is initialized from the brute force results. The mesh is refined at low resolution (about 1000×1000 pixels), again using the blob enhanced images. This stage is meant to capture the large scale deformations. Figure 12 on the next page illustrates this.

Figure 12: refined slice to slice registration of blob enhanced images



The transform is refined again, this time using the CLAHE enhanced images, at a higher resolution. This stage is meant to capture the local distortions. The results are illustrated in figure 13 on the following page.

Figure 13: refined slice to slice registration of CLAHE enhanced images



4.5 Stacking slices

Once all the slice pairs are registered, the slice to slice transforms are cascaded to map into the space of the target (fixed) slice. Any slice can be the target slice. Currently we use the first slice as the target.

The volume is assembled from fully overlapping regions, meaning that every pixel in any slice maps inside every other slice in the stack. In other words, every pixel must have a neighbor pixel in all the slices above and below. Pixels which do not satisfy this condition are cropped from the volume. This is illustrated in figure 14 on the next page.

Cascading transforms introduces geometric distortion artifacts in the slices far removed from the target slice. The slice to slice registration may stretch and squeeze local regions in the warped slice in order to improve the match with the fixed slice. When the slice to slice registration transforms are cascaded, the stretching/squeezing may become extreme. Figure 15 on page 19 demonstrates this.

The slice to slice registration is almost good enough for tracking individual features through the volume. Figure 16 on page 20 shows 12 cropped cross sections from the slice volume, where some features are readily recognizable in all the sliced.

Figure 14: the first slice in the volume

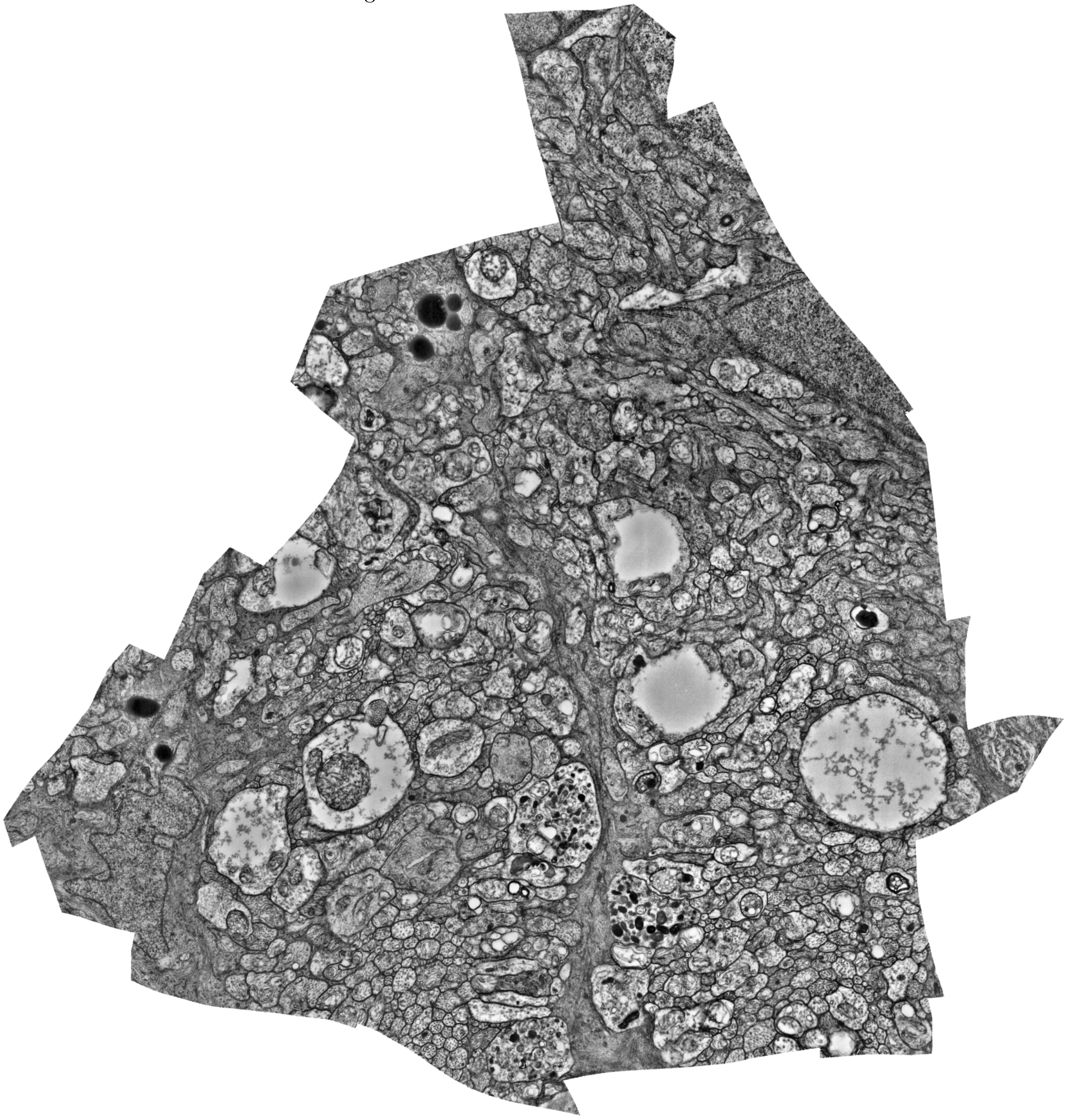


Figure 15: the last slice in the volume

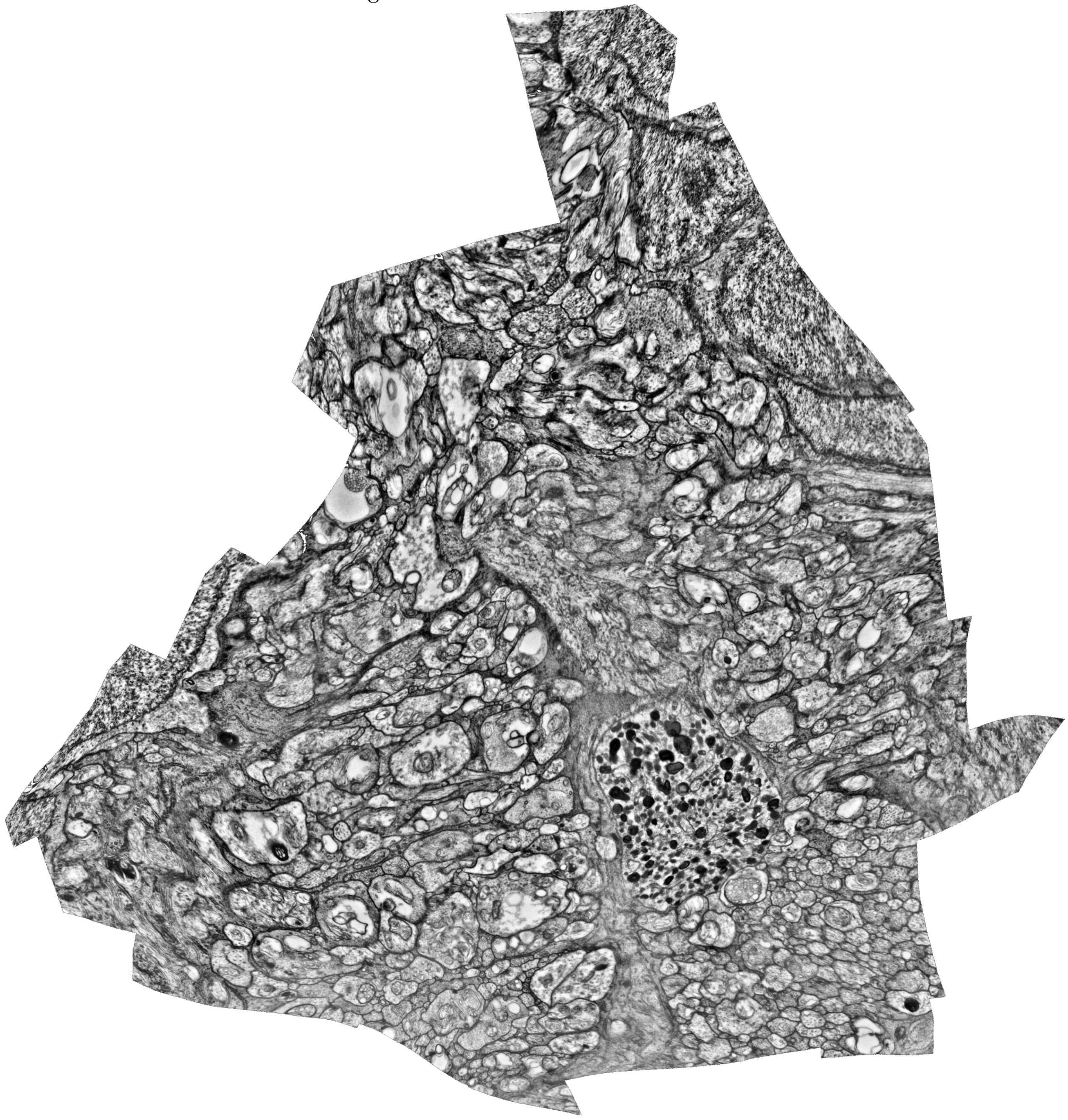
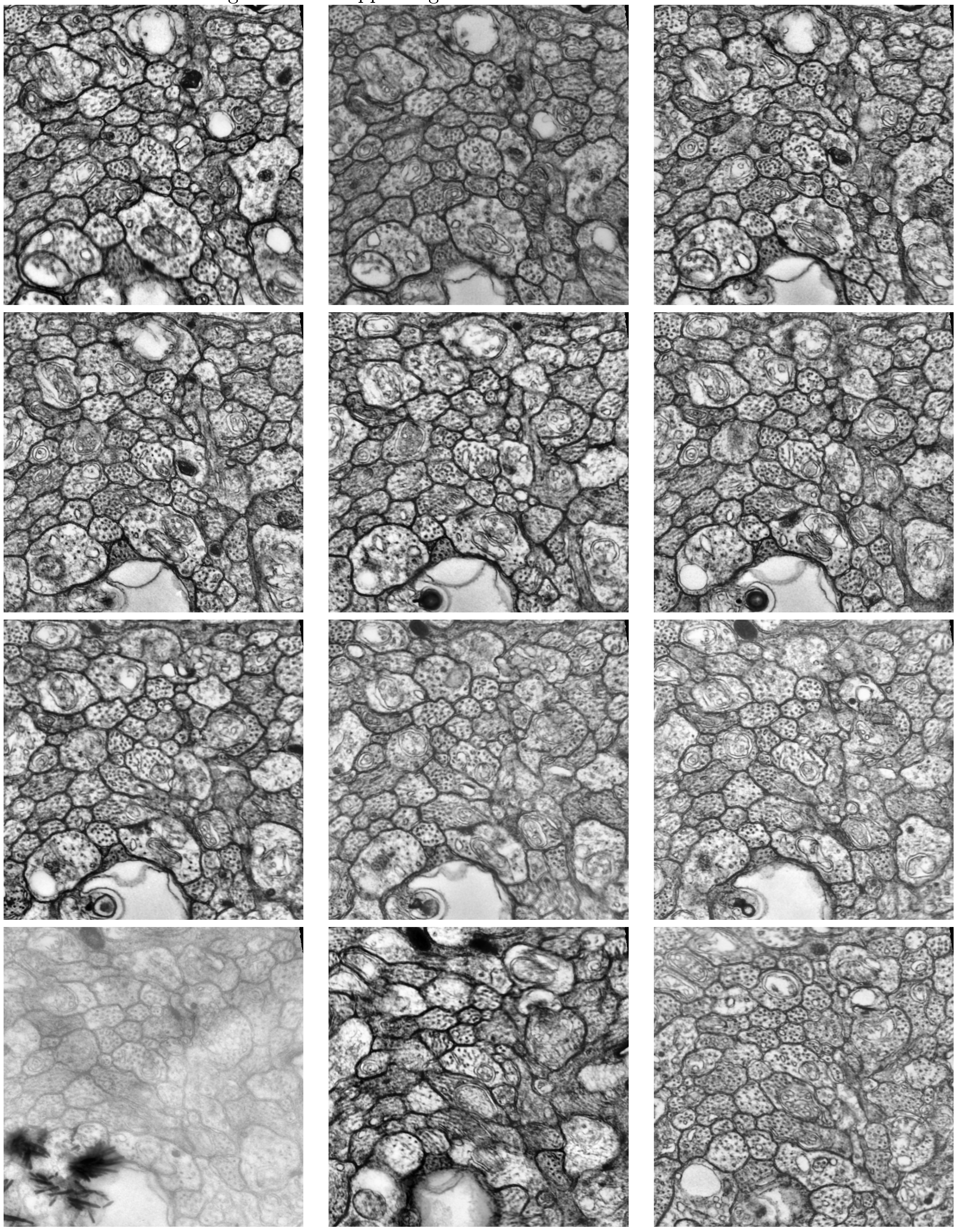


Figure 16: a cropped region from the stacked slice volume



References

[1] Girod, B. and Kuo, D. 1989. Direct estimation of displacement histograms. In Proceedings of the Optical Society of America Meeting on Understanding and Machine Vision, 73–76.

[2] Newton-Raphson Method for Nonlinear Systems of Equations. Numerical Recipes in C, second edition, 379–382.

[3] NLM Insight Segmentation & Registration Toolkit, http://www.itk.org/

[4] Fastest Fourier Transform in the West, http://www.fftw.org/