Implementation of an automatic image registration tool

Paul Koshevoy
koshevoy@cs.utah.edu

June 30, 2005

1 Motivation

The goal of this project is to provide a fully automatic tool for image registration and mosaicking of several hundred high-resolution images. This tool is primarily aimed at researchers working with electron microscopy images. A microscope rarely has a large enough field of view to cover the area of interest to the scientist with reasonable detail. Therefore, the area of interest has to be imaged in several tiles, following some overlapping tile pattern. The original area of interest is later reconstructed by laying out the image tiles into a mosaic. One problem particular to the microscopy images arises from the fact that the microscope introduces radial distortion into the image. Thus, even if the exact layout is known for the image tiles, the tiles may not match perfectly in the overlap region. When the number of tiles is more than just a few, the task of laying out the mosaic quickly becomes daunting, and is a prime candidate for automation.

2 Problem statement

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

- Find pairs of matching tiles.
- Deduce a tile ordering and build a rough estimate of the mosaic without radial distortion correction.
- Iteratively refine the mosaic by alternating the refinement of the radial distortion correction and position of each tile in the mosaic.

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 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-correlation Φ_{10} 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 auto-correlation terms $\Phi_{00} = F_0 \times F_0^*$ and $\Phi_{11} = F_1 \times F_1^*$ are used to enhance the cross-correlation term as follows

$$P = \frac{\Phi_{10}}{\sqrt{\Phi_{00} \times \Phi_{11}} + \epsilon}$$

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. Therefore, it is entirely possible that it is not necessary to enhance the cross-correlation term by canceling out the geometric average of the auto-correlation, and it may be used directly as $P = \Phi_{10}$. However, the current implementation of the mosaicking application follows exactly the technique described by Girod and Kuo.

The inverse Fourier transform of the cross-correlation

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

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 usually very noisy. Also, the PDF of two mismatched images may contain several maxima, or none at all. 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 Fourier domain, where it corresponds to a multiplication by a low-pass filter

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

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

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

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 Fourier domain as follows

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

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 strong 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 radial 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 radial 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 radially 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 coordinates. 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

$$\begin{aligned} T_{00} &= \begin{bmatrix} x_{max} \\ y_{max} \end{bmatrix} \\ T_{10} &= \begin{bmatrix} x_{max} - width\,(S_0) \\ y_{max} \end{bmatrix} \\ T_{01} &= \begin{bmatrix} x_{max} \\ y_{max} - height\,(S_0) \end{bmatrix} \\ T_{11} &= \begin{bmatrix} x_{max} - width\,(S_0) \\ y_{max} - height\,(S_0) \end{bmatrix} \end{aligned}$$

The current implementation of the application chooses the best permutation based on the normalized squared image differences metric. This metric is calculated as the sum of squared pixel differences within the overlap region, divided by the area of the overlap region. 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 Deducing the tile ordering

The image tiles have to be laid out in a particular order, such that each successive tile overlaps one of the previously laid out tiles. Matching tiles by definition have an overlapping area; consequently, prior to deducing the tile ordering 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 squared image differences metric mentioned earlier. 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

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

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

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

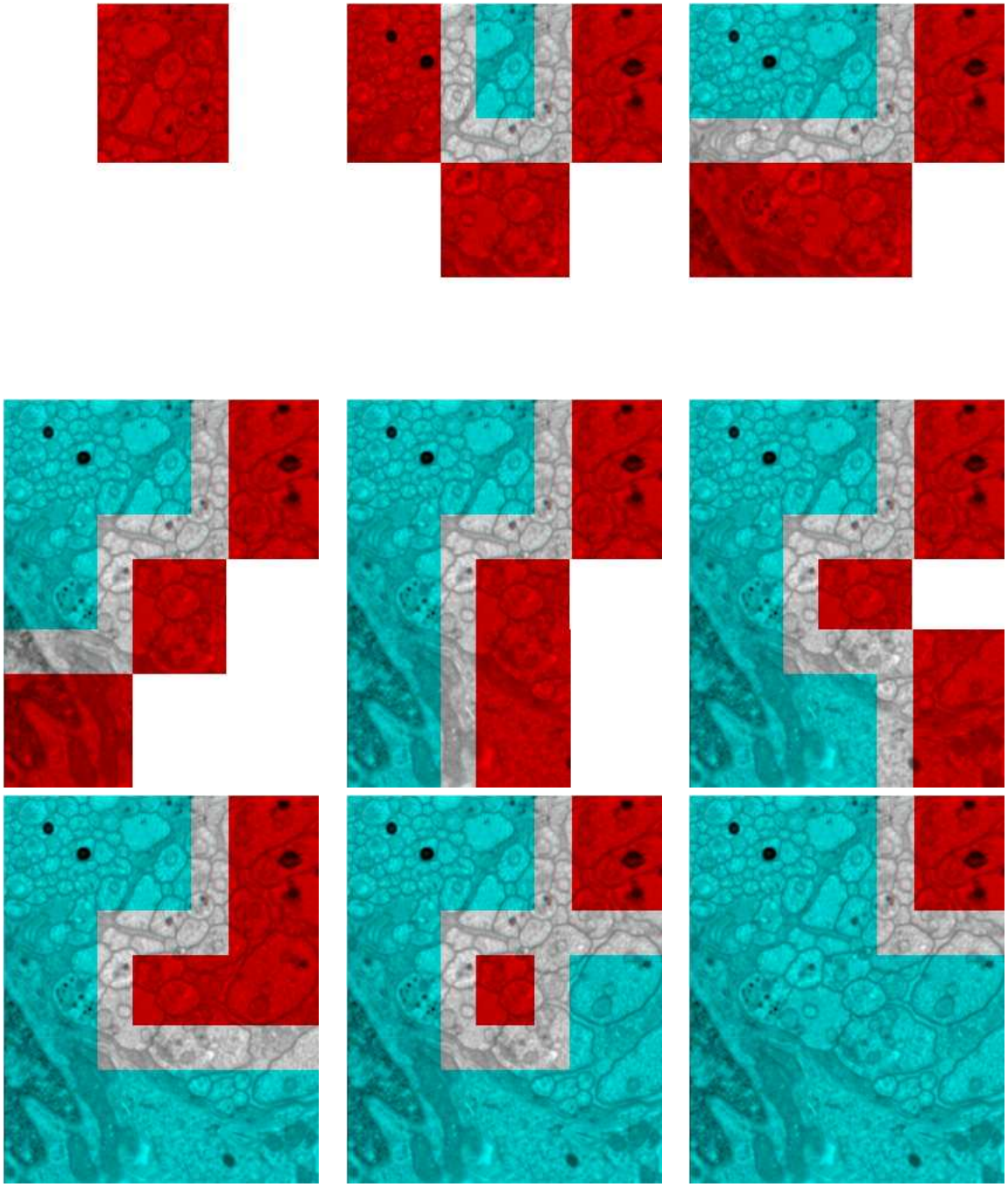
In order to generate the mosaic, it is necessary to select the target tile into which every other tile will be mapped. This is done by considering the total cost of the target tile candidates. The total cost is calculated as the cumulative cost of the mapping from the target tile to every other tile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

It is important that each tile added to the mosaic overlaps with a significant portion of one or more tiles that were added to the mosaic previously; otherwise ITK will not be able to refine the initial mosaic registration. Therefore, the tiles must be sorted in the order in which they will be added to the mosaic. It is not appropriate, however, to sort the tiles in the left-to-right and top-to-bottom order because such ordering may fail to provide successively adjacent overlapping tiles. The sorting algorithm currently implemented in the mosaicking application optimizes the area of overlap of the successive tiles.

The sorting starts out with the mosaic target tile, under the assumption that the target tile overlaps several neighbors with low mismatch error. The target tile is added to the perimeter and marked as matched. The algorithm iterates until the perimeter is empty. At each iteration, a tile is removed from the perimeter and appended to a list of previously considered perimeter tiles. Following, the overlap area of the perimeter tile and every other unmatched tile is evaluated, and the results are stored in a list and filtered. The filtering is a simple thresholding algorithm that discards tiles which overlap the perimeter tile less than half as well as the best tile in the list. The filtered list is sorted in the ascending order and added at the head of the perimeter list. This step is necessary in order to ensure that the tiles with the greater area of overlap are given higher consideration priority while maintaining the adjacency of successive tiles. The tiles listed in the filtered list are marked as matched in order to be removed from further consideration, and the loop repeats. Once the perimeter list is empty, the list of previously considered perimeter tiles specifies exactly the order in which the tiles should be added to the mosaic.

Figure 1 on the following page demonstrates the evolution of the perimeter (shown in red) as the sorting algorithm progresses. The corresponding tile ordering is illustrated in figure 6 on page 10.

Figure 1: tile sorting algorithm perimeter illustration



3.3 Radial distortion correction

In order to correct for radial distortion, the tile has to be warped by a radial distortion transform. The transform is defined as follows

$$\begin{aligned} x(u,v) &= u_c + (u - u_c) \times S(u,v) \\ y(u,v) &= v_c + (v - v_c) \times S(u,v) \end{aligned}$$

$$
\begin{aligned}
S(u,v) &= \sum_{n=0}^{N-1} k_n \times \left(\frac{R(u,v)}{R_{max}} \right)^{2n} \\
R(u,v) &= \sqrt{(u-u_c)^2 + (v-v_c)^2}
\end{aligned}
$$

where $[u_c\, v_c]^T$ is the center of radial distortion. The transform is normalized by R_{max}. Thus, the radial distortion transform is parameterized by coordinates $u_c\, v_c$, normalization constant R_{max} and polynomial coefficients $k_0 ... k_{N-1}$. In order to simplify the computational burden, it is assumed that R_{max} corresponds to the maximum distance from the center of distortion to the corners of the tile. The location of the center of distortion is unknown, therefore it is assumed to be at the center of the tile. Additionally, the number of polynomial coefficients is limited to $N = 2$. Thus, only k_0 and k_1 are needed to define the transform.

The polynomial coefficients are found iteratively by the ITK[3] image registration framework. The ITK image registration framework consists of the following components

- Two images that must be matched (fixed image and moving image).
- A metric that quantifies the quality of the match between the images.
- A transform.
- An optimizer.

Although the image tiles are referred to as the fixed image and the moving image, the names are misleading. The ITK image registration framework actually computes the transform that maps (moves) from the space of the fixed image into the space of the moving image. Thus, the transform is the inverse of the mapping from the space of the moving image into the space of the fixed image. In order to calculate the dimensions of the mosaic, it is necessary to be able to transform points from the moving image space into the fixed image space. Since the corresponding transform is unavailable, the inverse mapping is calculated numerically via Newton's method[2].

Within the ITK image registration framework, the optimizer manipulates the parameters of the transform in order to achieve the best possible match between the images. Currently, the $itk :: LBFGSBOptimizer$ class is used. This optimizer implements the "Limited memory Broyden Fletcher Goldfarb Shannon minimization with simple bounds". It was chosen because it is reasonably fast and allows the user to impose bounds on the parameter values. The current implementation of the application takes advantage of this capability to clamp k_0 to the range $[0.9, 1.1]$ and k_1 to $[-0.1, 0.1]$. This restricts the range of radial scaling to $\left[k_0^{min} - k_1^{max}, k_0^{max} + k_1^{max}\right]$.

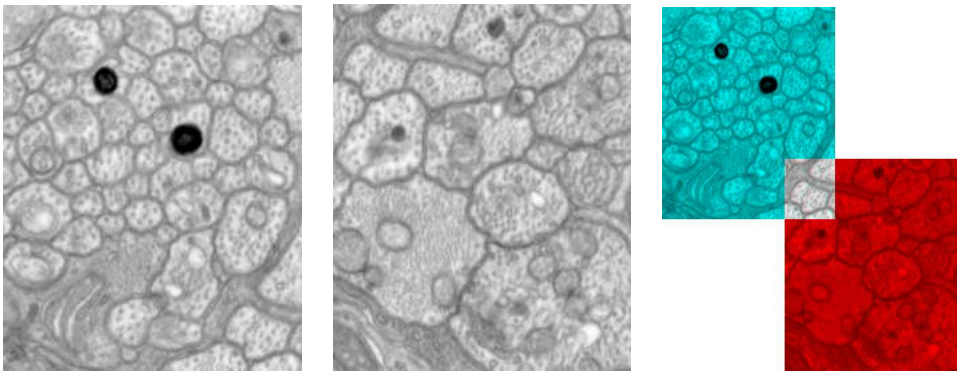
The image registration iterates until it converges or exceeds the maximum number of iterations (specified by the user). The resulting transform parameters define a radial distortion transform which best matches the fixed and moving images by correcting for the radial distortion present in either of the images.

4 Demonstration of the correctness of implementation

4.1 Tile matching

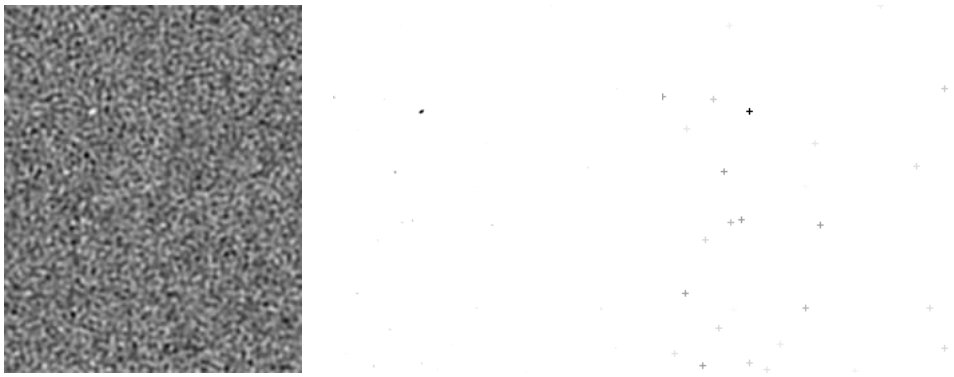
Figure 2 on the following page shows two matching image tiles. These tiles have undergone a mild radial distortion with parameters $k_0 = 0.95$ and $k_1 = 0.05$. The overlap area between these tiles is roughly 8%. Figure 3 shows the displacement PDF corresponding to these two tiles, as well as the isolated pixel clusters corresponding to the PDF maxima. There are a total of 19 maxima isolated in the PDF. Filtering the maxima leaves only one eligible maximum for consideration, which indicates that the tiles are well matched.

Figure 2: matching tiles



tile 0 tile 4 mosaic 0:4

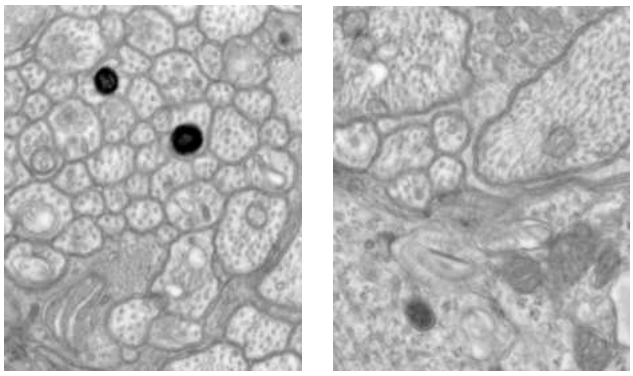
Figure 3: displacement PDF for matching tiles



PDF 0:4 maxima clusters PDF maxima

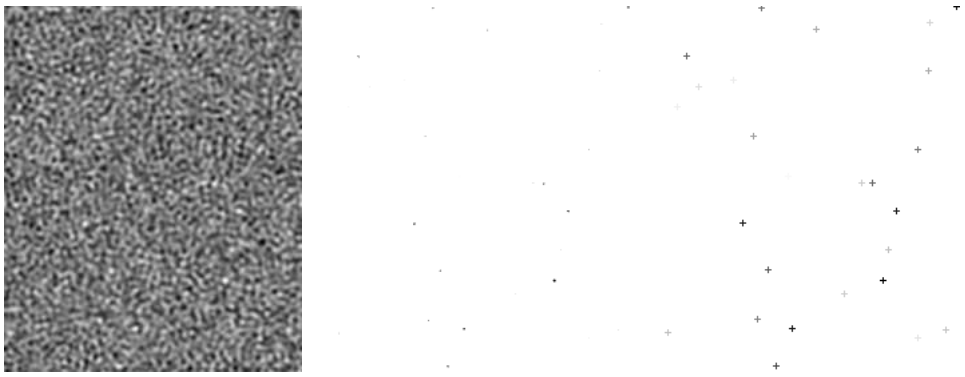
Figure 4 on the next page shows two mismatched tiles. Figure 5 shows the corresponding displacement PDF and PDF maxima. There are 26 maxima isolated in this PDF. After filtering there are still 11 maxima left. Ideally there would be only one maximum left, therefore this PDF indicates that the tiles do not match.

Figure 4: mismatched tiles



tile 0 tile 8

Figure 5: displacement PDF for mismatched tiles

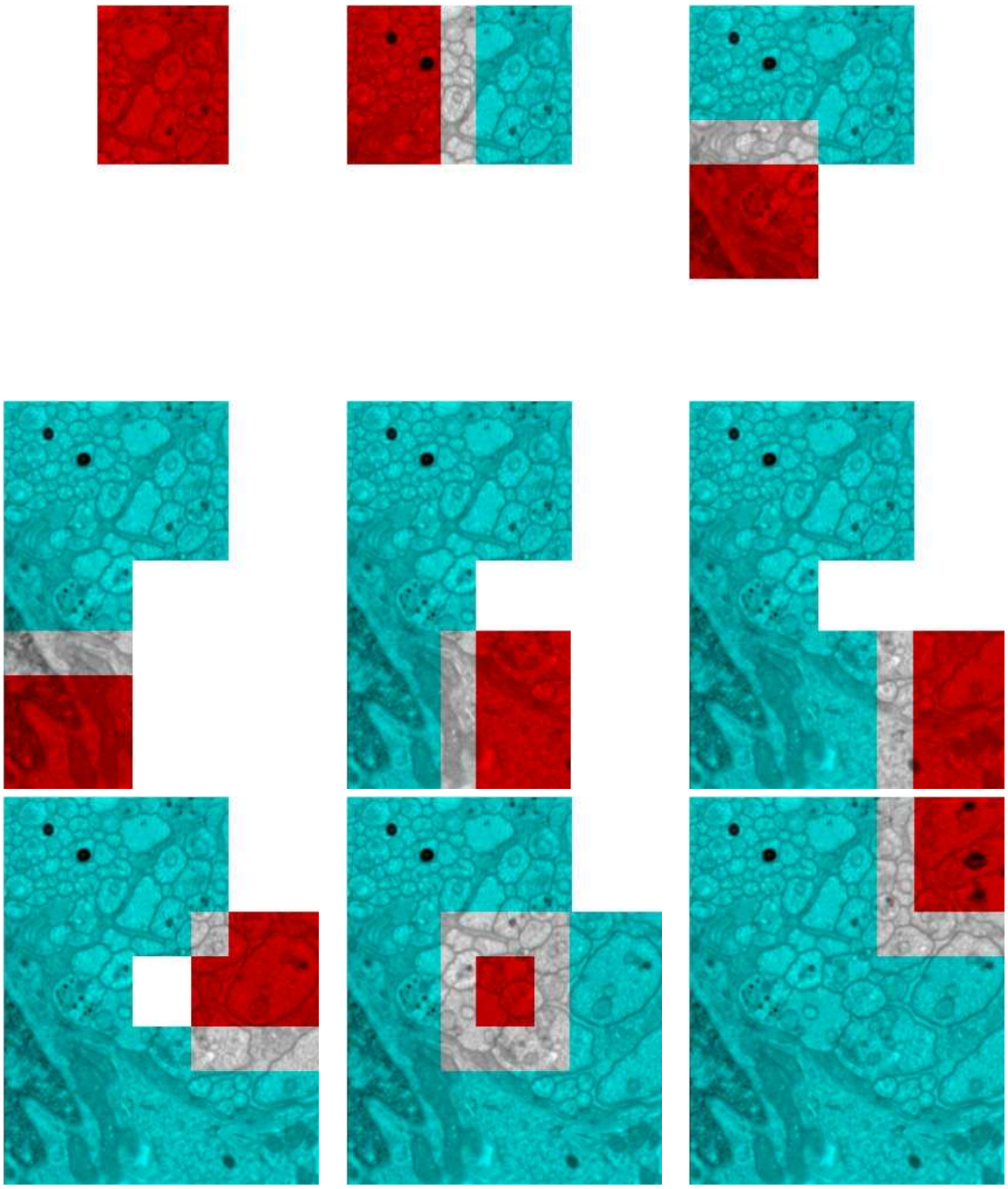


PDF 0:8 maxima clusters PDF maxima

4.2 Tile ordering

Figure 6 on the following page illustrates the order in which the tiles are added to the mosaic. As can be seen, the algorithm lays out the red tiles such that they have significant overlap with previous tiles (shown in blue).

Figure 6: tile ordering

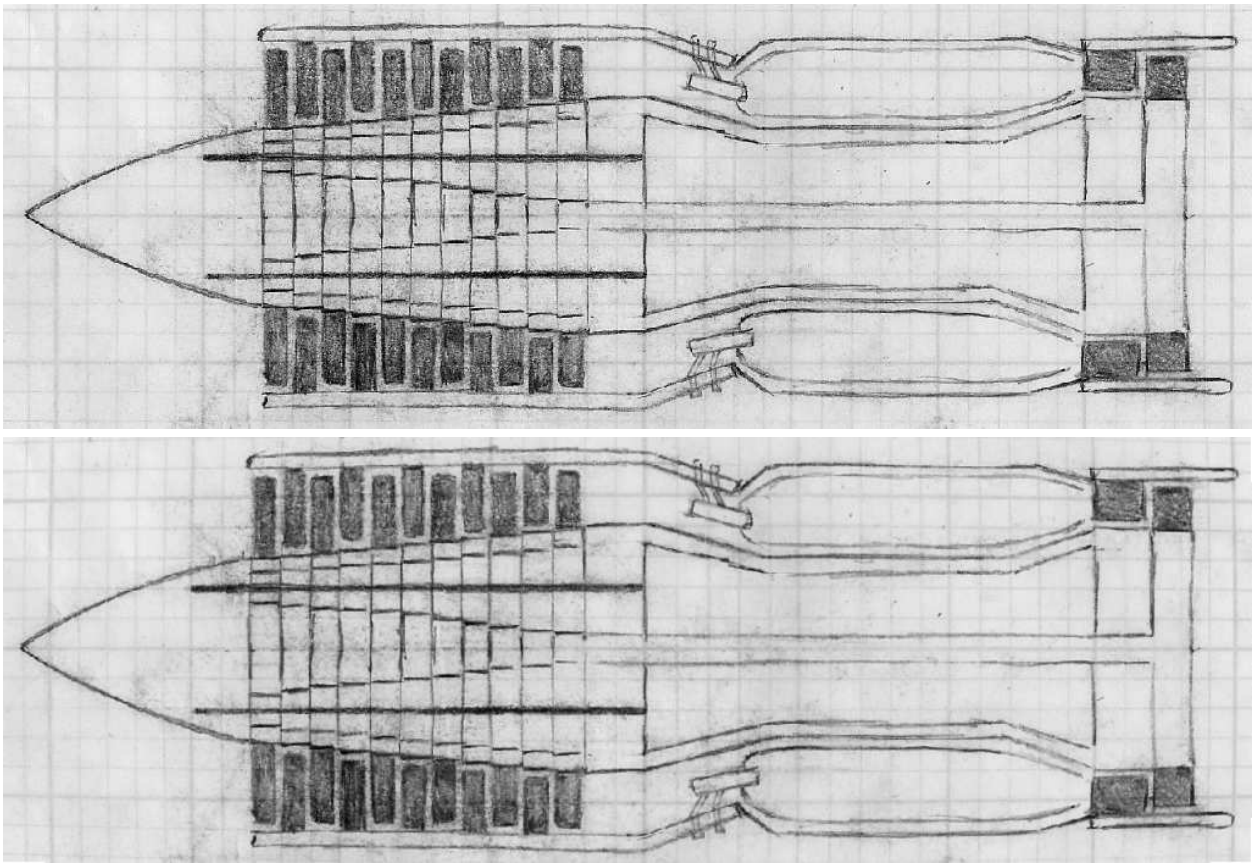


4.3 Radial distortion correction

Unfortunately, radial distortion correction is not fully functional in the current implementation of the application. Distortion correction between two images one of which is a radially distorted version of the other works well. However, radial distortion correction between partially overlapping images fails to find the correct parameter values k_0 and k_1. Figure 7 on the next page shows two images corresponding to the best case scenario for radial distortion correction. The image on the bottom was radially distorted with parameters

$k_0 = 0.95$ and $k_1 = 0.05$.

Figure 7: original and radially distorted image



The mosaic corresponding to these two images is shown in figure 8 on the following page. The image on the bottom demonstrates variance within the overlapping regions of the mosaic.

Figure 8: blurring and high variance due to radial distortion

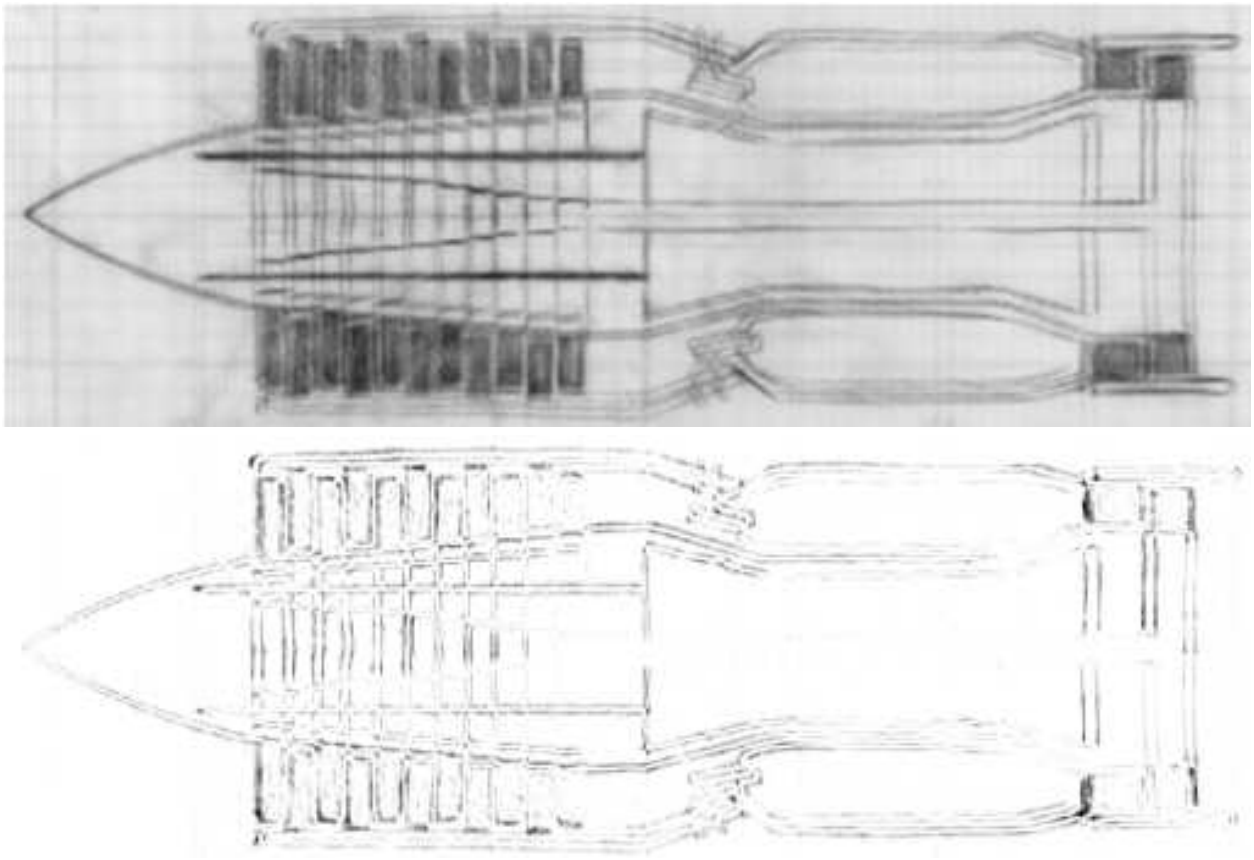
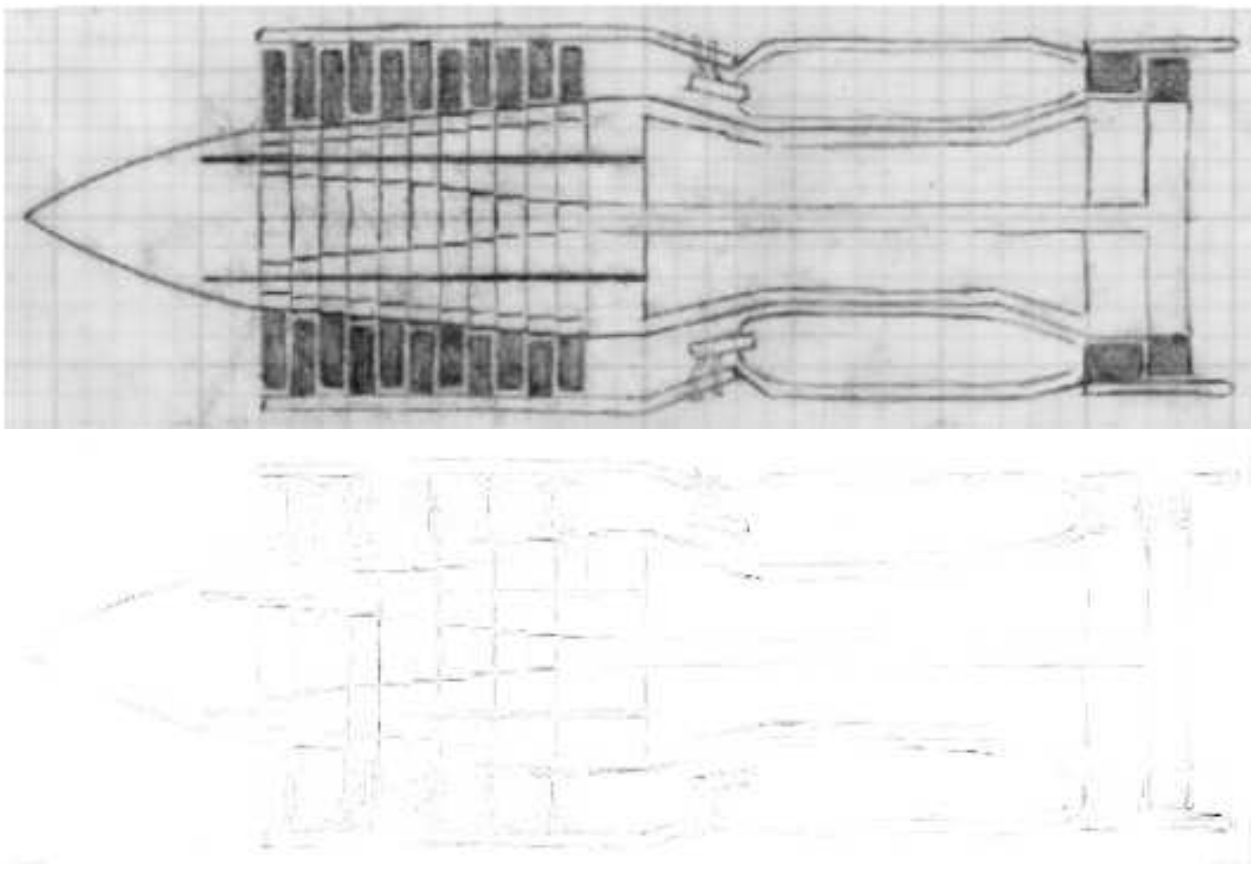


Figure 9 on the next page shows the results of iterative refinement of the displacement vector and radial distortion parameters for the radially distorted image. As can be seen, the variance within the mosaic has been significantly reduced. The optimizer was able to find parameters $k_0 \approx 1.05$ and $k_1 \approx -0.05$ to correct for the radial distortion in the distorted image.

Figure 9: low blurring and variance due to corrected radial distortion



5 Real world results

Figure 10 shows 12 tiles of one mosaic. These tiles were under exposed during development, yet the mosaicking application is able to assemble a mosaic in the correct order, as can be seen in figure 11.

Figure 10: sample electron microscopy tiles

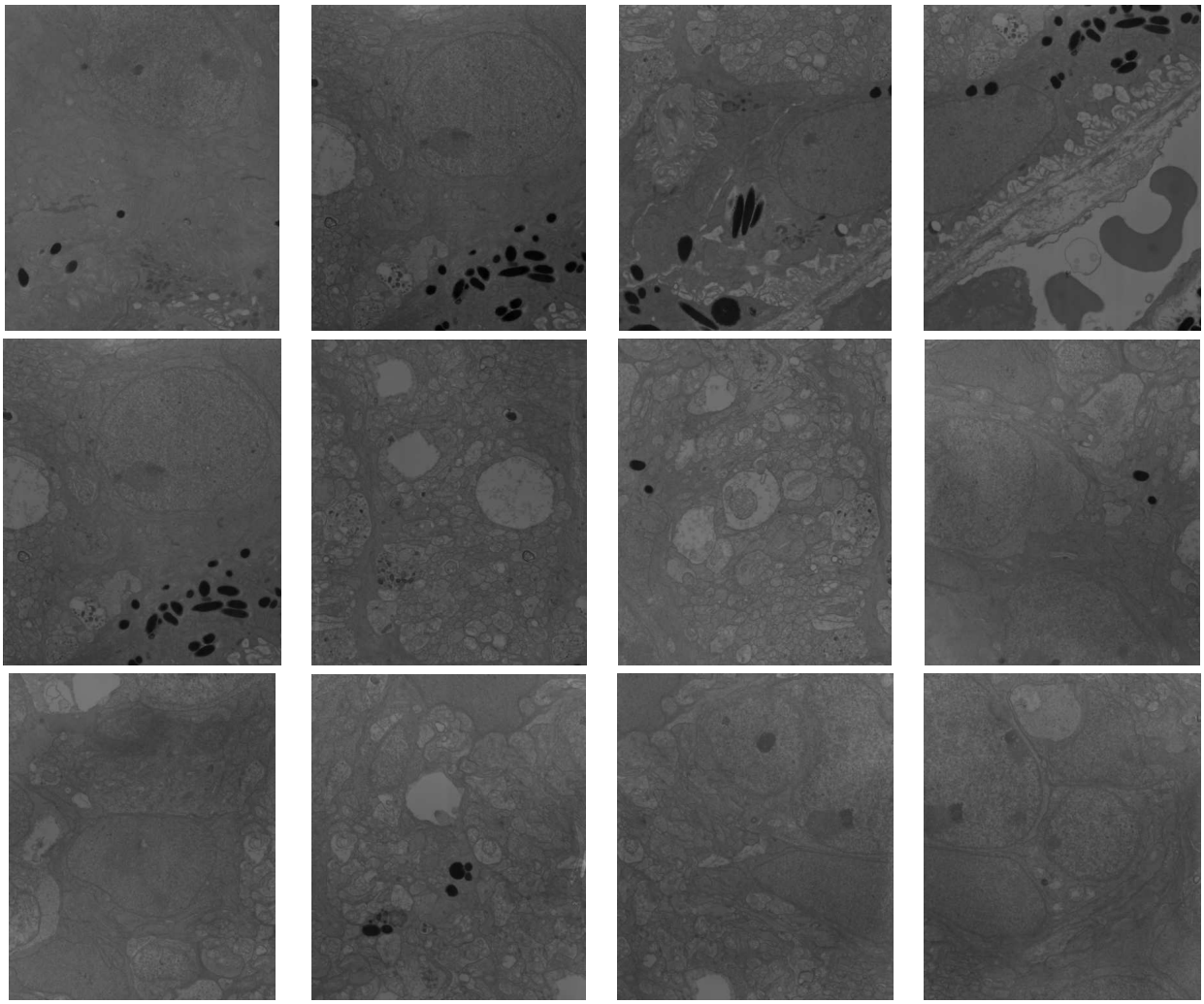
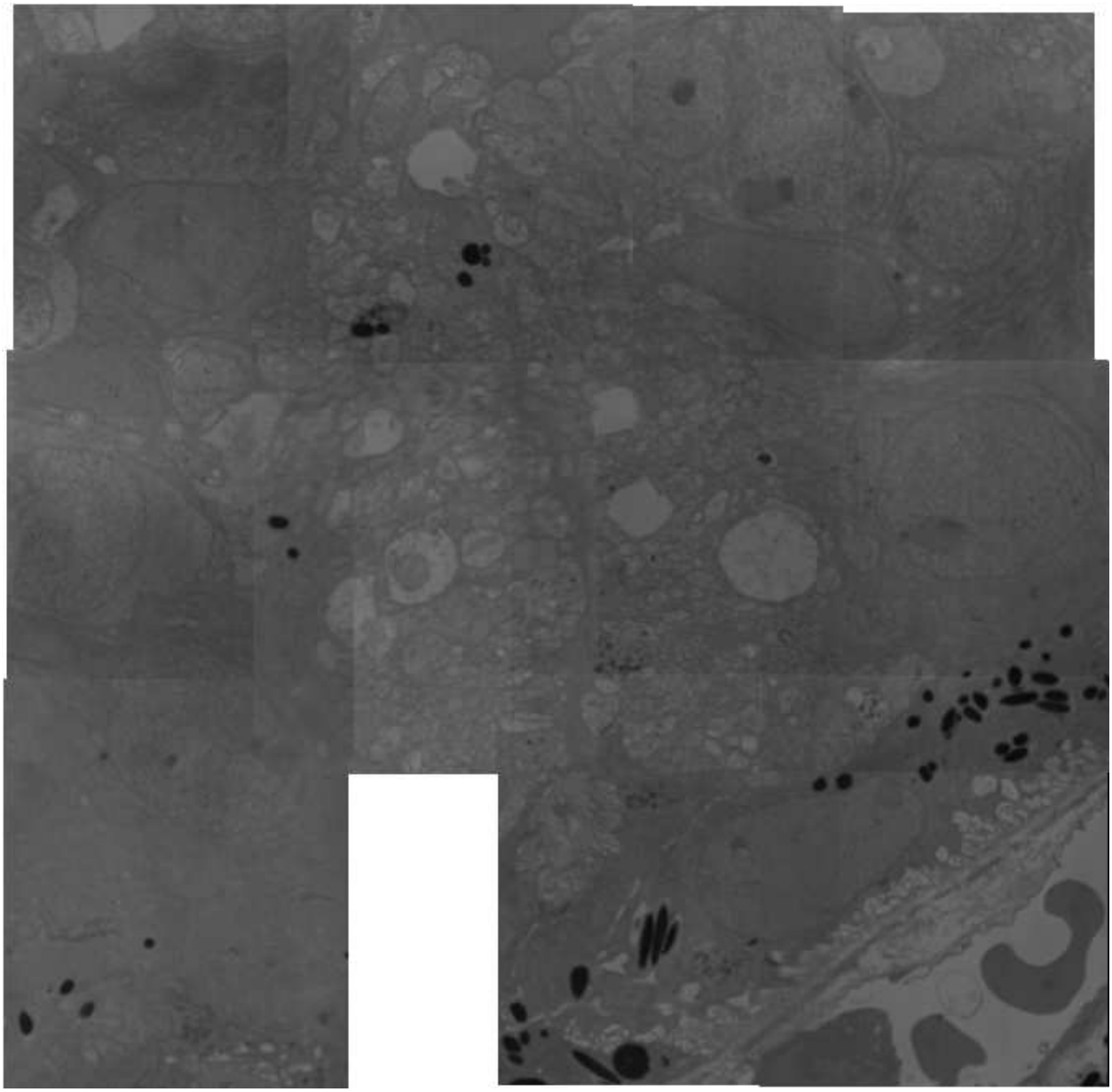


Figure 11: mosaic of 12 electron microscopy tiles



6 Summary and conclusion

As can be seen from the results presented in sections 4 and 5, the current implementation of the application is capable of robust mosaicking of overlapping image tiles.

Unfortunately, radial distortion correction still needs improvement. The current incapability of finding the correct distortion parameters based on a 30% to 20% overlap between adjacent tiles seriously limits the usefulness of the application to the wider scientific community. Naturally, various methods of improvement will be attempted in order to rectify this limitation.

One idea for improvement is already being considered. Since the electron microscopy tiles are typically produced by the same microscope, it follows that the tiles within one mosaic are all distorted by the same radial distortion parameters. Thus, if these parameters can be found for one tile, the rest of the tiles can be

undistorted automatically.

Another idea for improvement that is currently being considered is the global refinement of the mosaic, bypassing the ITK image registration framework and optimizing the tile transforms all at once in order to minimize the variance within the overlapping tile regions of the mosaic.

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/