Report for Exercise 2 and 3

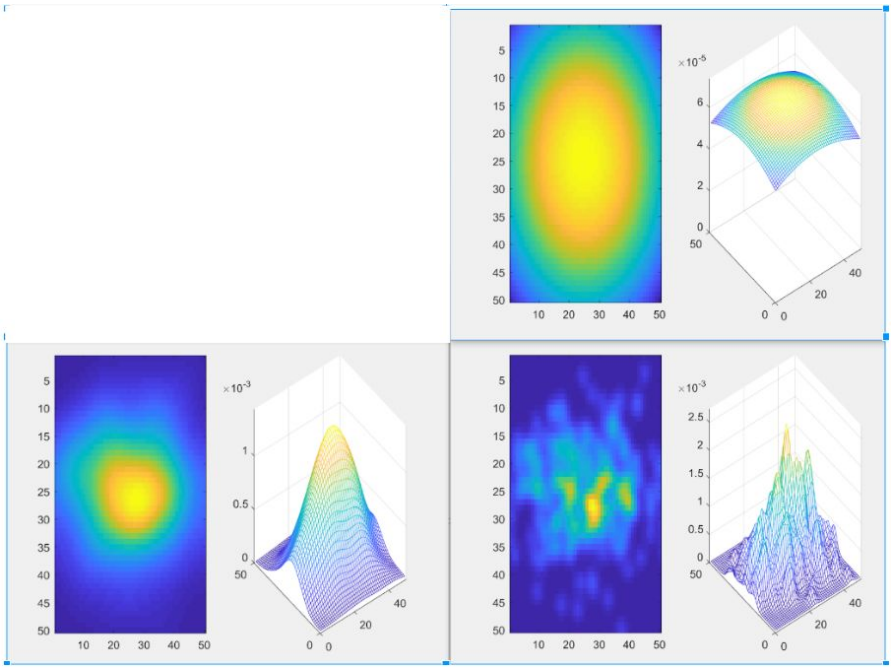
Gruppe

Phillip Brinck Vetter s144097

Jacob Jon Hansen s134097

Scale space is very usable for image analysis, as we often have objects of interest with different scale in the image. We can have images of leaves up close or images of leaves in a tree from a far away, same object but at completely different scales.

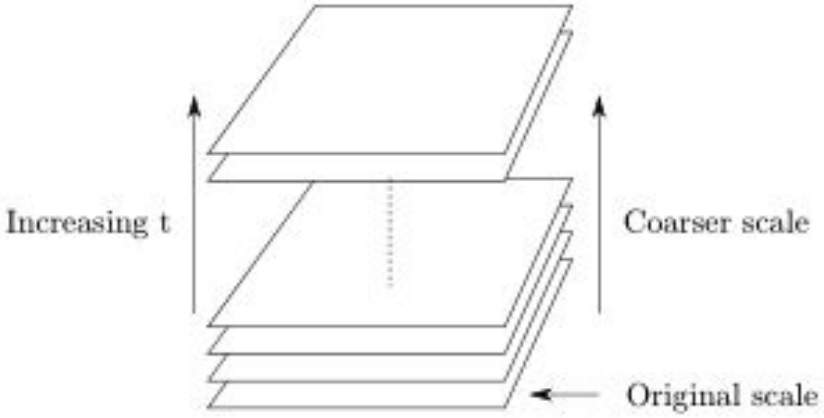
Related to Exercise 2



Above is an example of smoothed intensity image, different smoothing parameters will give different results so we can either focus on small high frequency features or larger scales where we have bigger features. We do local max/min suppression to find the location of such features.

Edges = very big gradient in image so we want min/max of first derivative to find these

Blops = very big curvature, so we want second derivative to find these. I.e blops will be big and then steadily go down in a smoothed image, edges will be more abrupt.


Increasing t
Coarser scale
Original scale

The Gaussian

The scale-space is based on convolution with the gaussian defined over the region given by the scale parameter t as follows: (note that we should do at least 3 sqrt(t) which equals 3 standard deviations 99.7% of area of the convolution filter).

$$x \in I = \left[-5\sqrt{t}\,,\, 5\sqrt{t}\right]$$

Blob Detection Synthetic Images

Scale T = 1

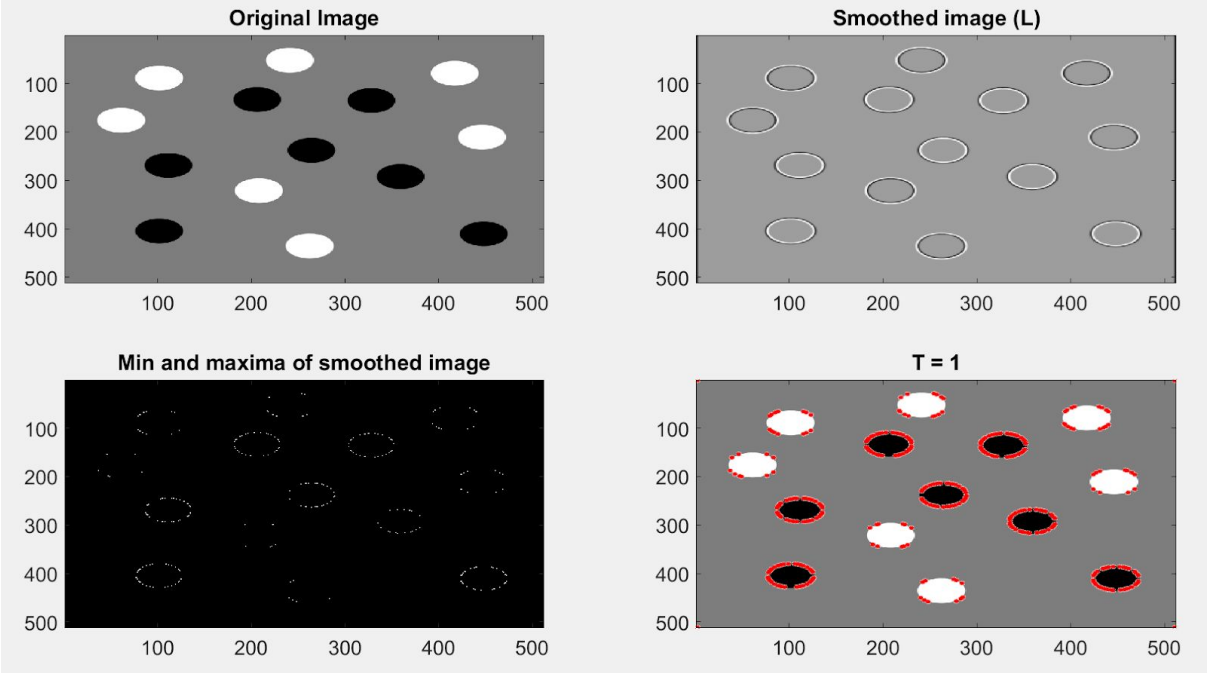


Figure 1: The above image shows the smoothing process for the test_blob_uniform.png. 1) Original Image. 2) Laplacian Image. 3) Regional maximum and minimum in the Laplacian Image after thresholding. The detections along the blob-edges are found because of the low scale used. 4) Circles outlining the detected blobs as shown in Min-Max Image.

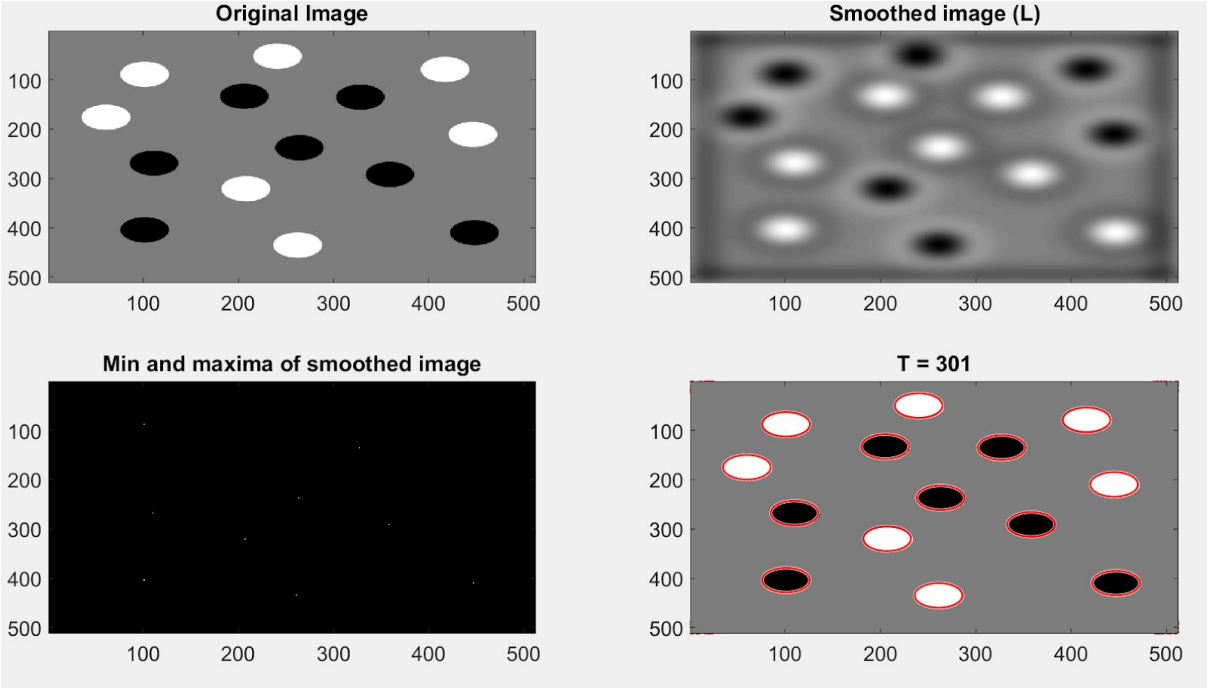


Figure 2: Image above is similarly to that shown in **Figure 1** on the previous page. It is evident from the Laplacian Image (2) that the smoothing is much stronger, and as such the scale at which blobs are detected matches the the size of the blobs in the image. The Max-Min Image (3) reveals that only the blob centers remains once the regional maximum and minimum in the Laplacian have been thresholded. The circles outlining the blobs in (4) now match exactly as intended.

Scale T = 11, 61, 151, 791

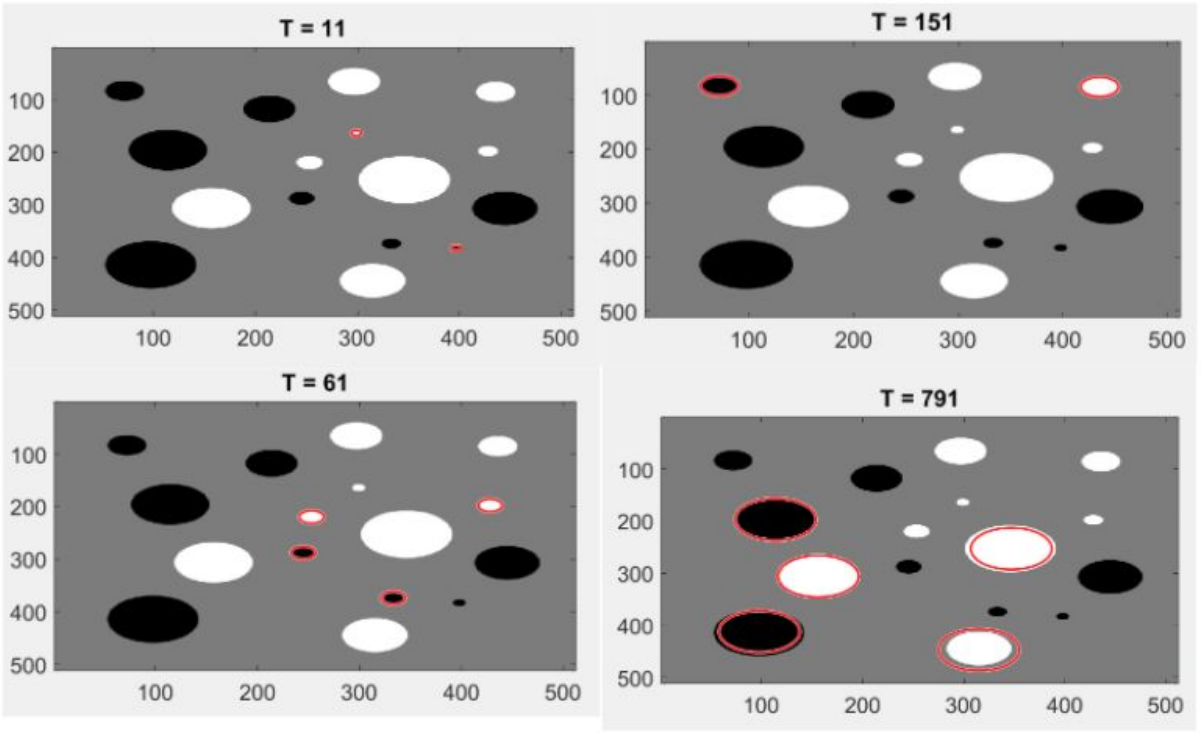


Figure 3: The procedure described in **Figure 1** and **Figure 2** shown for various scales. It is evident that blobs of a specific size are only detected when the associated scale is applied.

Multiple Scale Detection

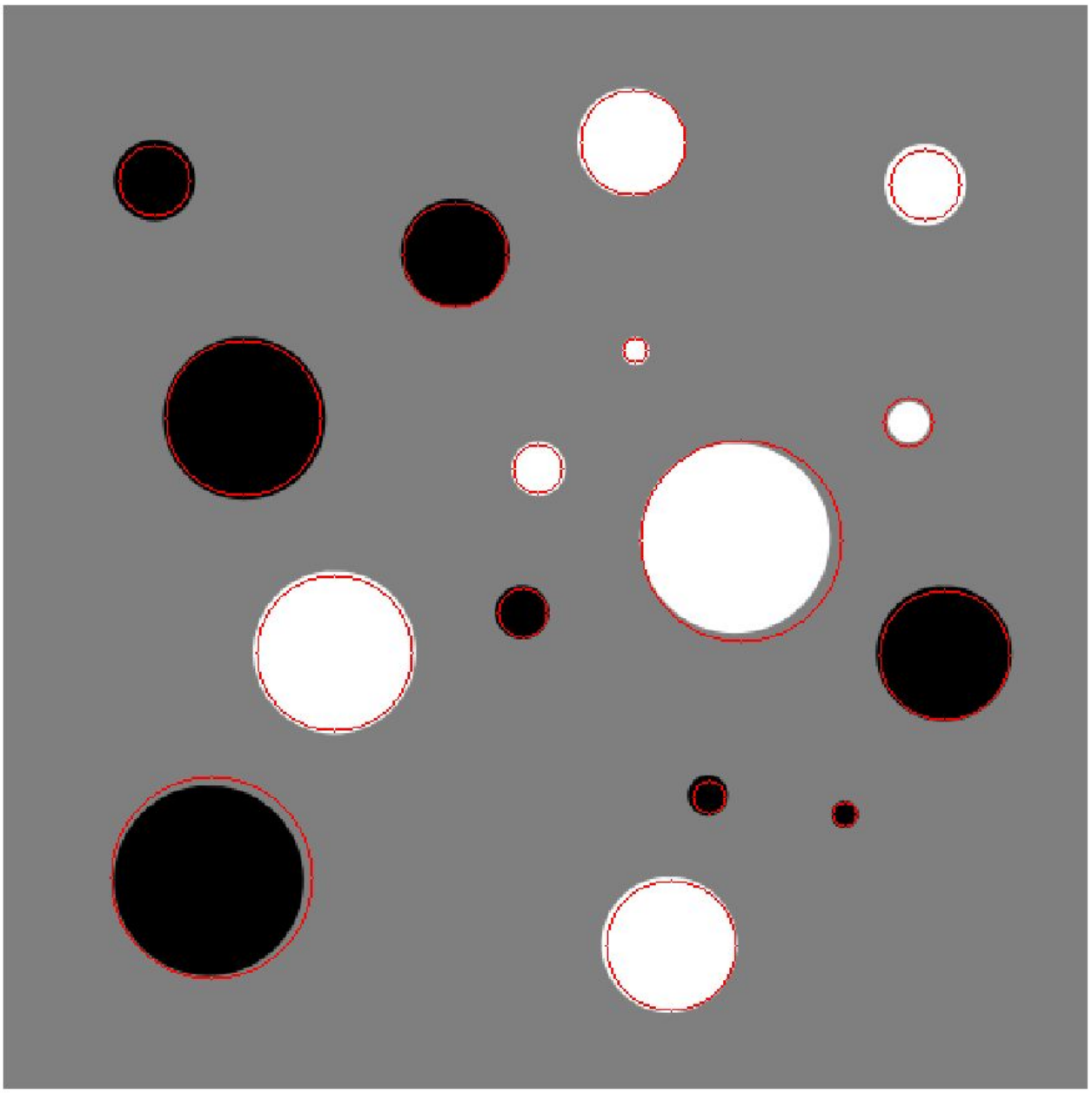


Figure 4: Blob detection over multiple scales by finding the maximum in a 3x3x3 neighbourhood of the scale-space. The algorithm was implemented using Python.

Blob Detection for Fiber Images

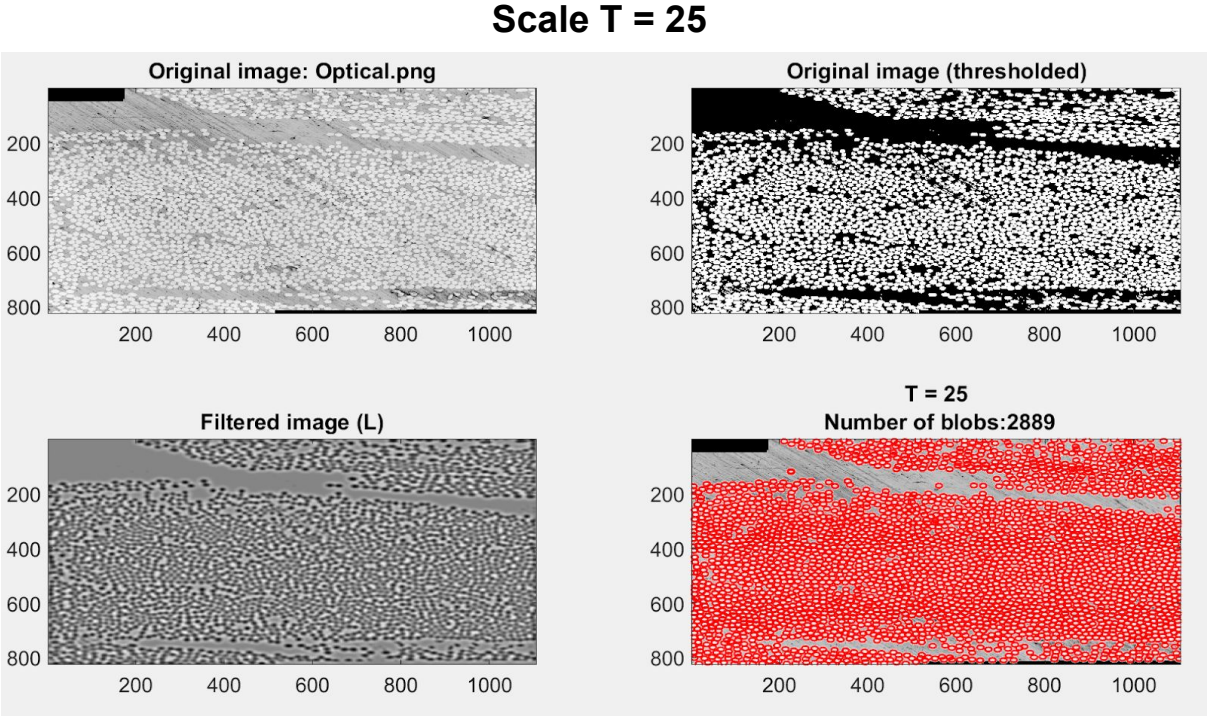


Figure 5: The blob detection algorithm applied to the Optical.png image. An additional initial thresholding was done as it was deemed to improve the detection. Since the fibers are all the same size only a single scale was needed. The number of detected blobs are shown in (4) to be approximately 2900. This is also shown in the figure below.

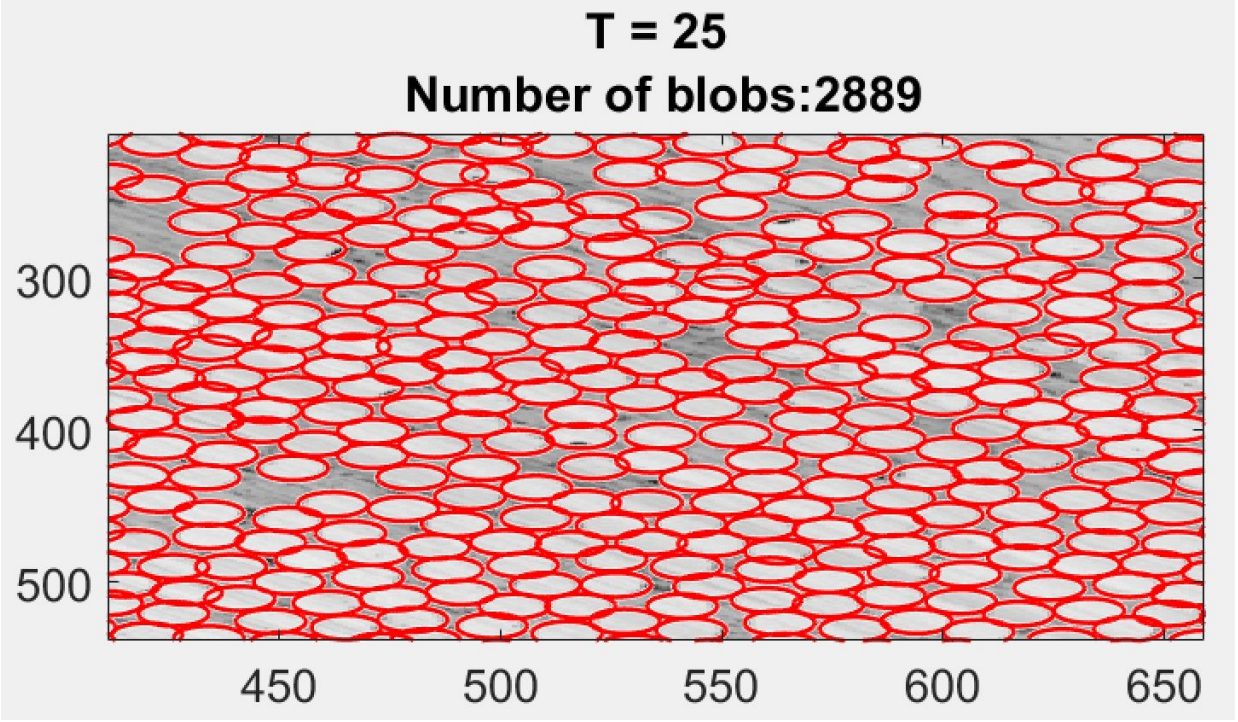


Figure 6: Near view that shows the work of the algorithm in greater detail. The detections are in great correspondence with the actual size of the fibers.

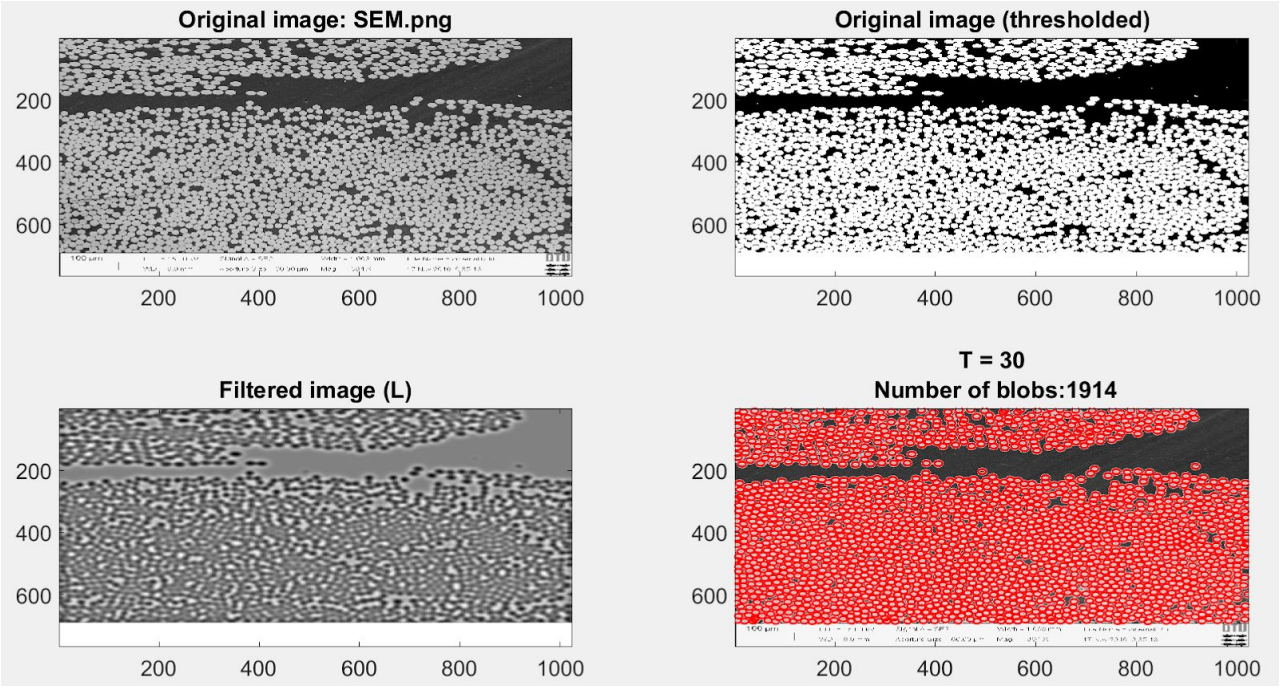


Figure 7: Similar detect as that described in **Figure 6** for the image SEM.png. The chosen scale is T = 30 and the number of detected blobs were roughly 1900.

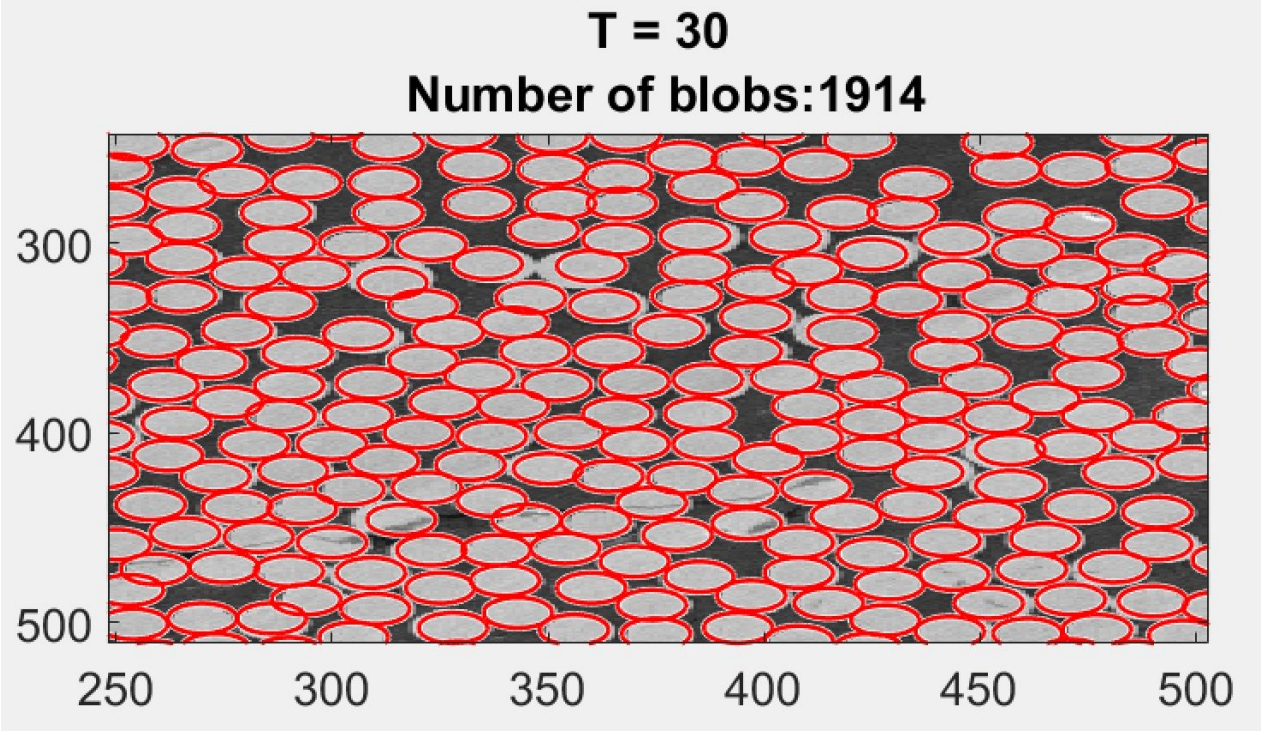


Figure 8: Near view that shows the work of the algorithm in greater detail. The detections are in great correspondence with the actual size of the fibers, even though a few of the detections are off by a bit.

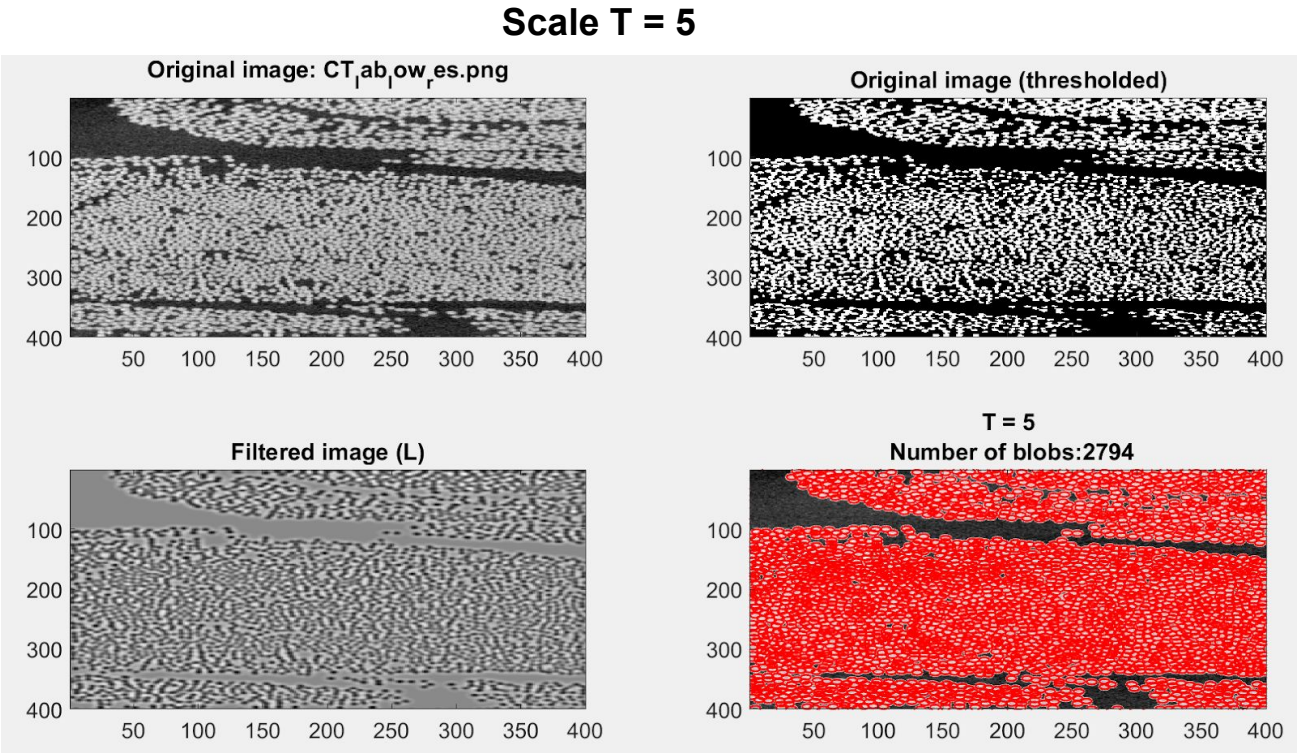


Figure 9: Blob detection for the image CT_lab_low_res.png. The most correct scale was found to be T = 5, and its seen from (4) that the number of blobs found was roughly 2800.

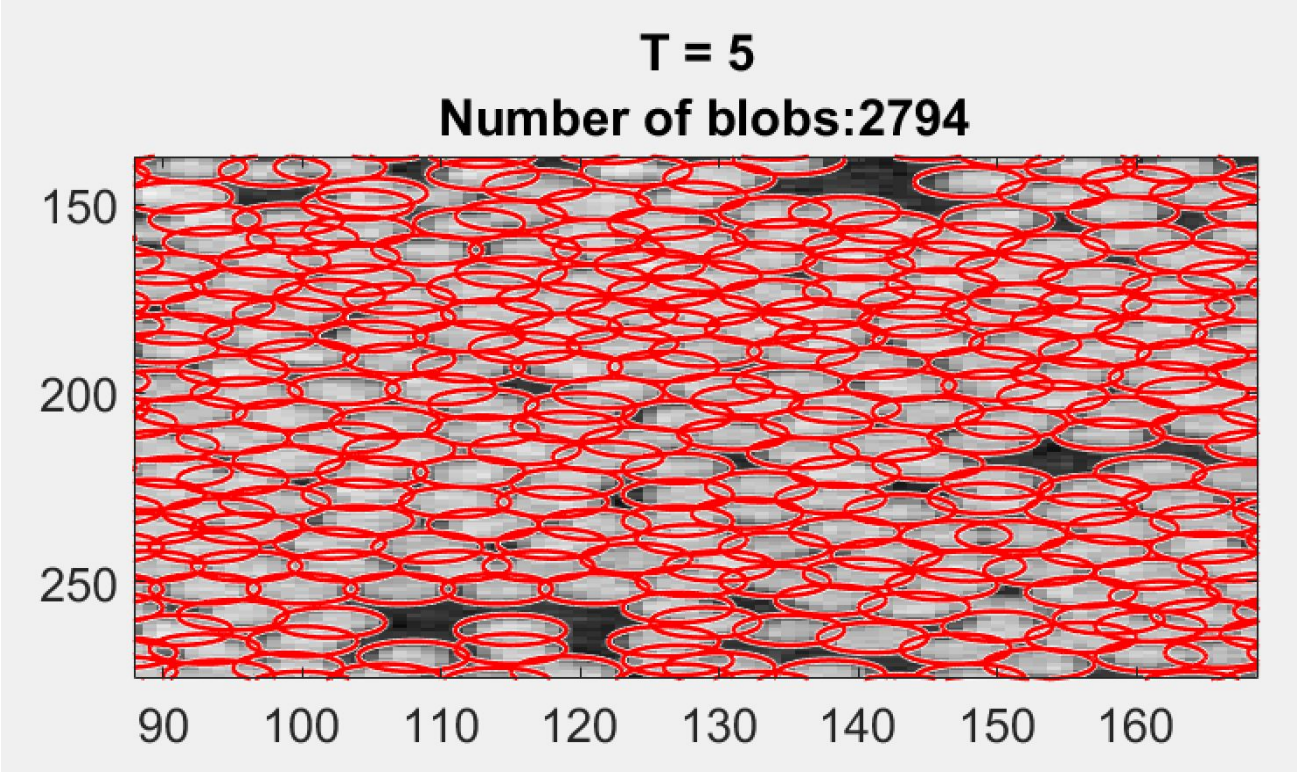


Figure 10: Zoomed view of a region of (4) in **Figure 9**. The algorithm seems to capture the size and amount of fibers in general, however a few wrongly placed blobs or miscounts is not unlikely to happen. This is of course also enhanced by the low image quality.

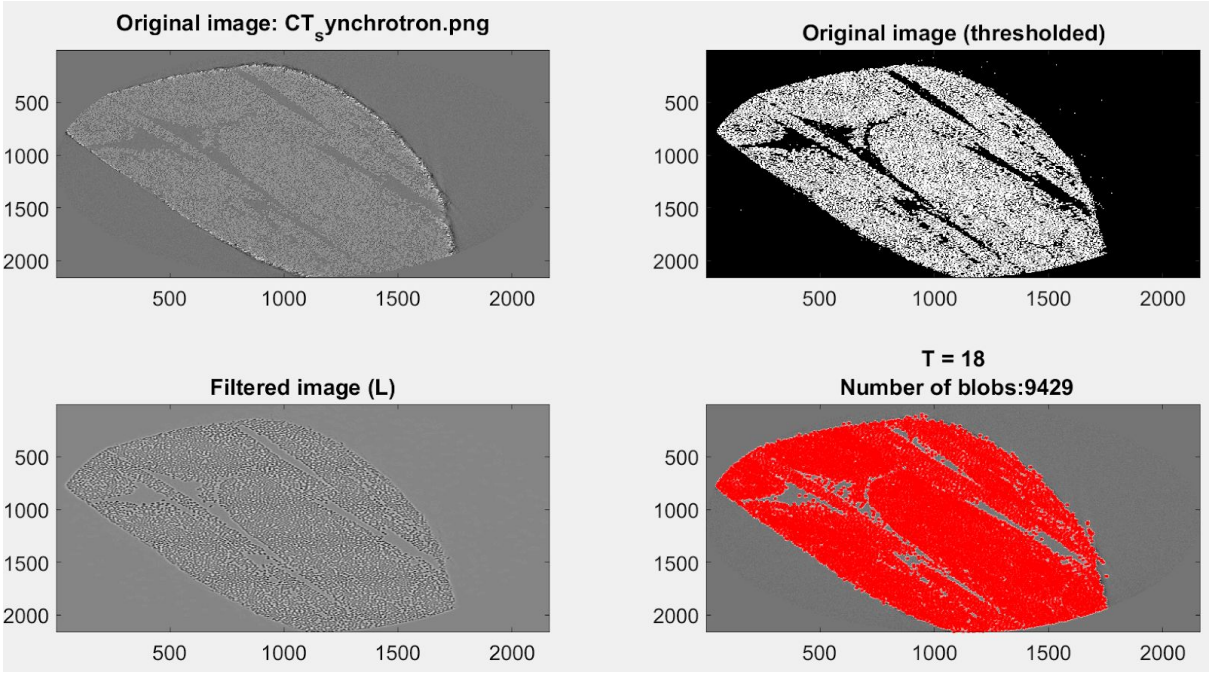


Figure 11: Blob detection for the CT_synchrotron.png image. The chosen scale was T = 18 and the amount of blobs found were roughly 9500.

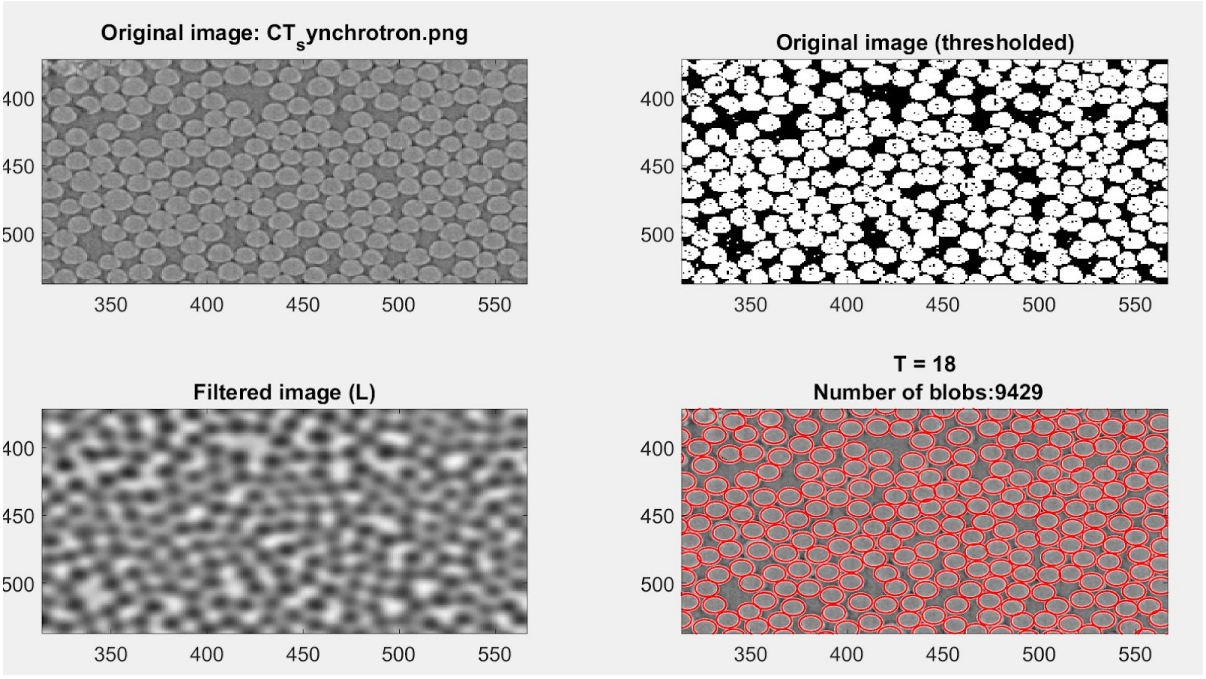


Figure 12: A closer look at the images presented in **Figure 11.** Judging from (4) it is clear that the algorithm works well on the high resolution image and detects the blobs correctly without problems.

Related to Exercise 3

SIFT:

- **Constructing a scale space** This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".
- **LoG Approximation** The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created earlier.
- **Finding keypoints** With the super fast approximation, we now try to find key points. These are maxima and minima in the Difference of Gaussian image we calculate in step 2. Sub-pixel location can be found with taylor expansion.
  5 blurred images → 4 DOG→ 2 max/min extrema images
- **Get rid of bad key points** Edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used here. (Threshold and gradient, edge one big one small, flat both small, corner both big)
- **Assigning an orientation to the keypoints** An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.
- **Generate SIFT features** Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features. Lets say you have 50,000 features. With this representation, you can easily identify the feature you're looking for (say, a particular eye, or a sign board). That was an overview of the entire algorithm. Over the next few days, I'll go through each step in detail. Finally, I'll show you how to implement SIFT in OpenCV!

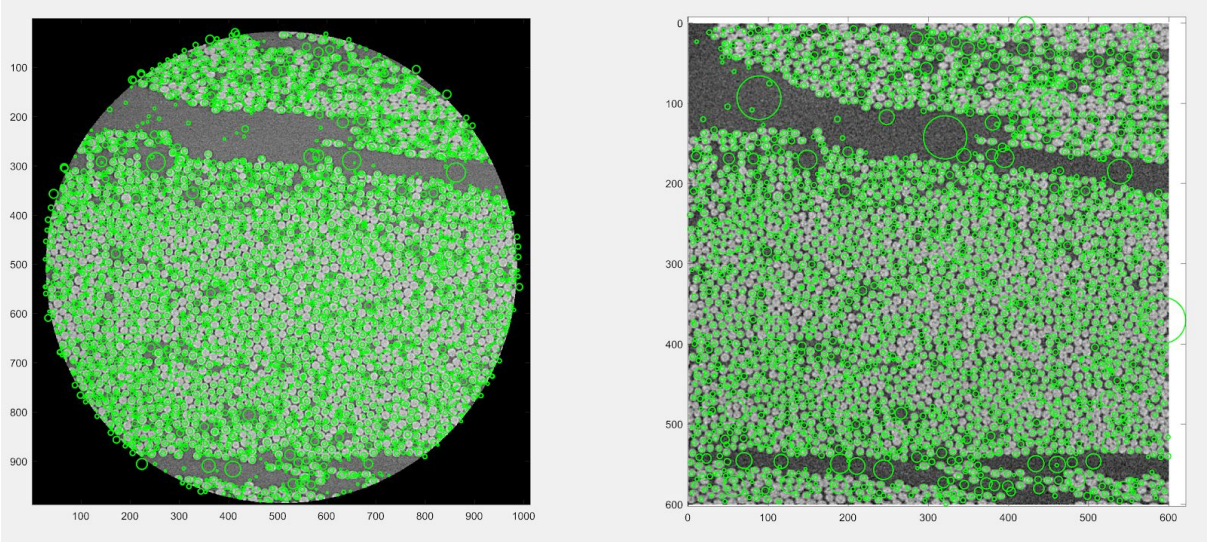


Figure 13: On the two two images above (Left: CT_lab_high_res.png, Right: CT_lab_med_res.png) computed SIFT features using the *vl_sift* command from *vlfeatroot* toolbox are shown as green circles.

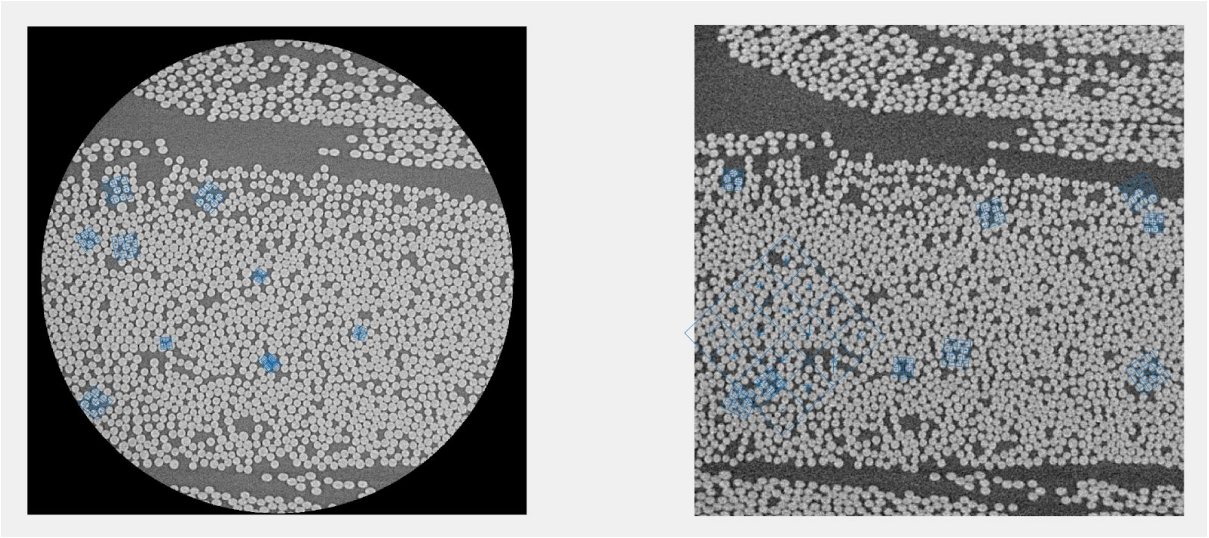


Figure 14: Overview of the descriptor histograms produced from the *vl_sift* command. The locations of the blue marks seen in the two images corresponds to a key feature, while a closer look at the marks would reveal the actual eight-binned 4x4 histogram. The histogram can be seen in **Figure 15**

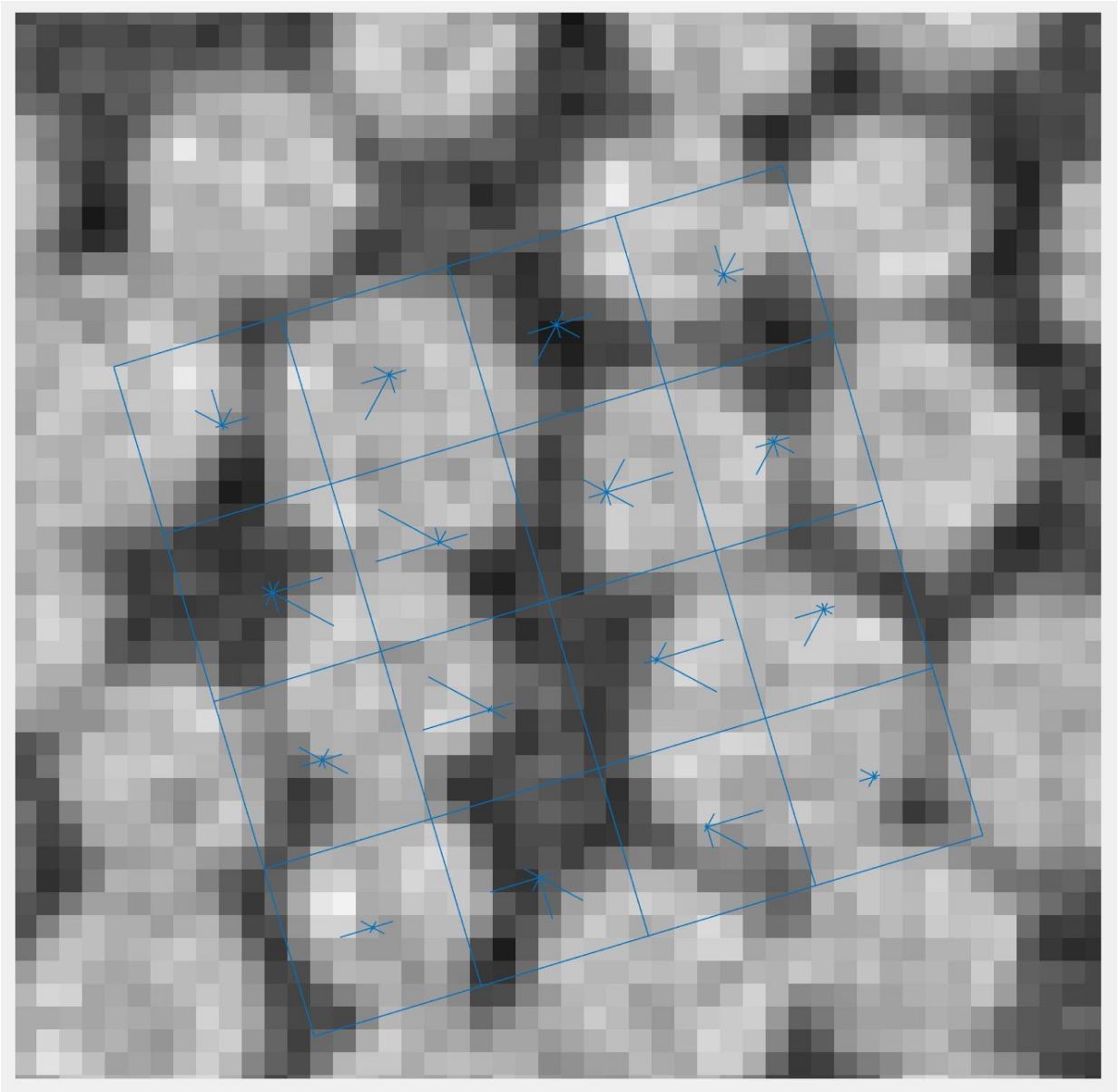


Figure 15: Descriptor histogram for a selected key feature as seen in **Figure 14**. 8 bin histogram, 16 of them 8x16=128

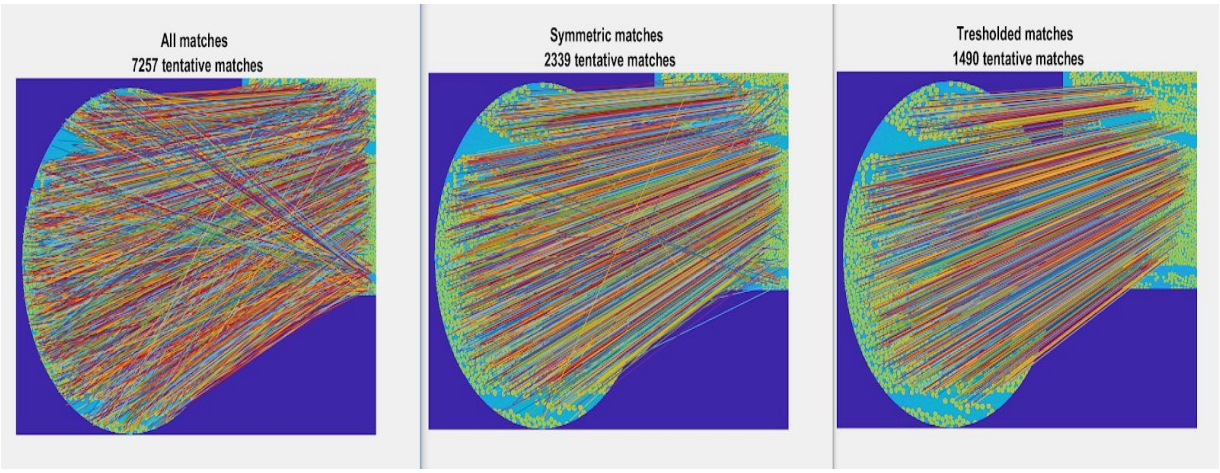


Figure 16: The three images above show the three different approaches to matching features. In (1) all matches are accepted. In (2) we require symmetry between the matches resulting in a threefold reduction. In (3) Lowe's similarity match is used with a threshold of 0.6. It is evident that the threshold

$$\eta(\delta_{I_i}, \delta_{\hat{I}}) = \frac{d_1(\delta_{I_i}, \delta_{\hat{I}_j})}{d_2(\delta_{I_i}, \delta_{\hat{I}_k})}$$

method seems to fare the best. - Measures uniqueness, that is the distance between the best matches and the distance between the 1 and 3. best match must be greater than some threshold → Ensures uniqueness of descriptor

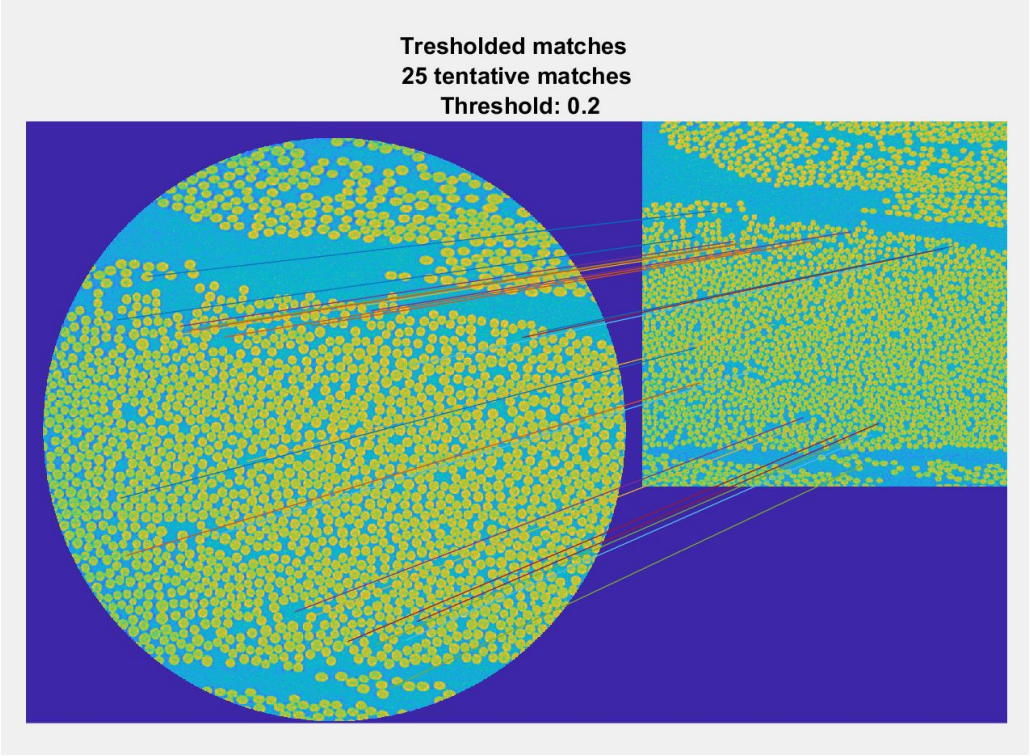


Figure 17: The Lowe's similarity matching with a threshold of 0.2. The high demand results in only the strongest 25 matches remaining.

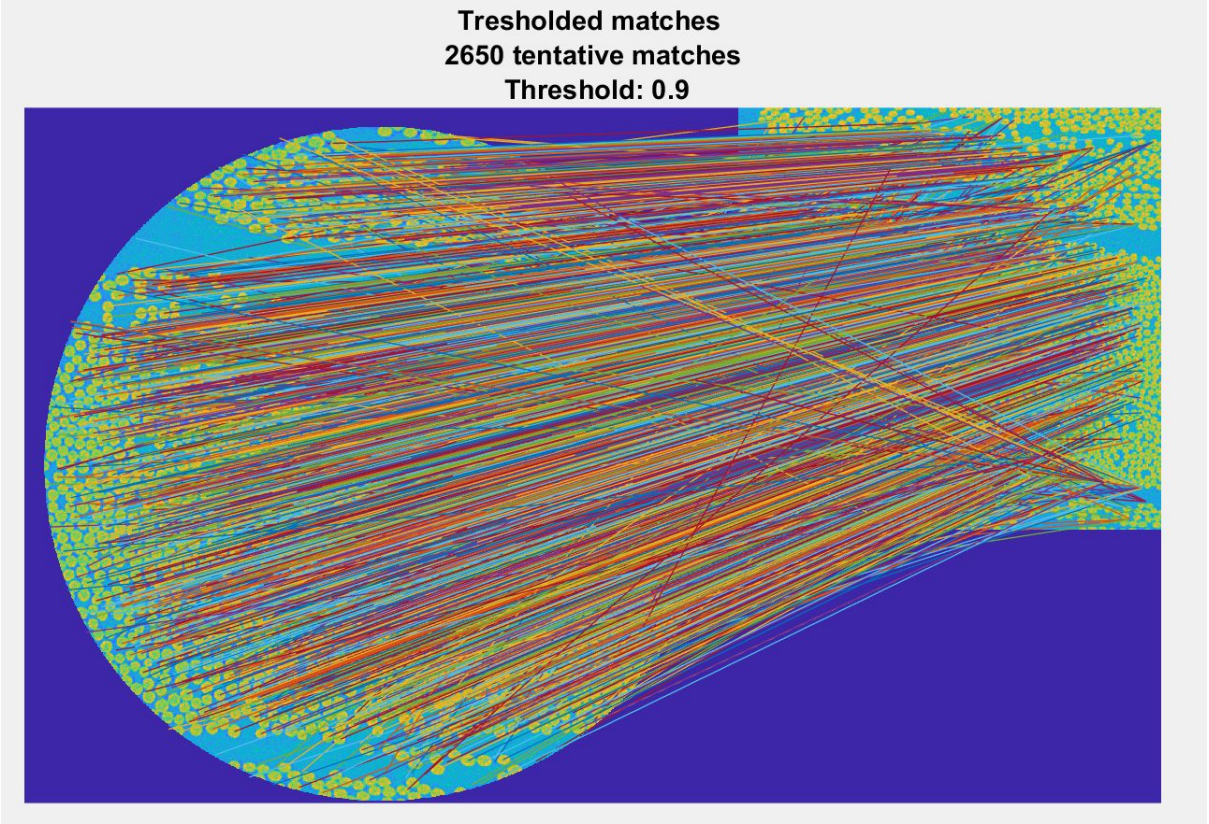


Figure 18: The Lowe's similarity matching with a threshold of 0.9. It is seen that the resulting allowed matches surpass that of the symmetry approach.

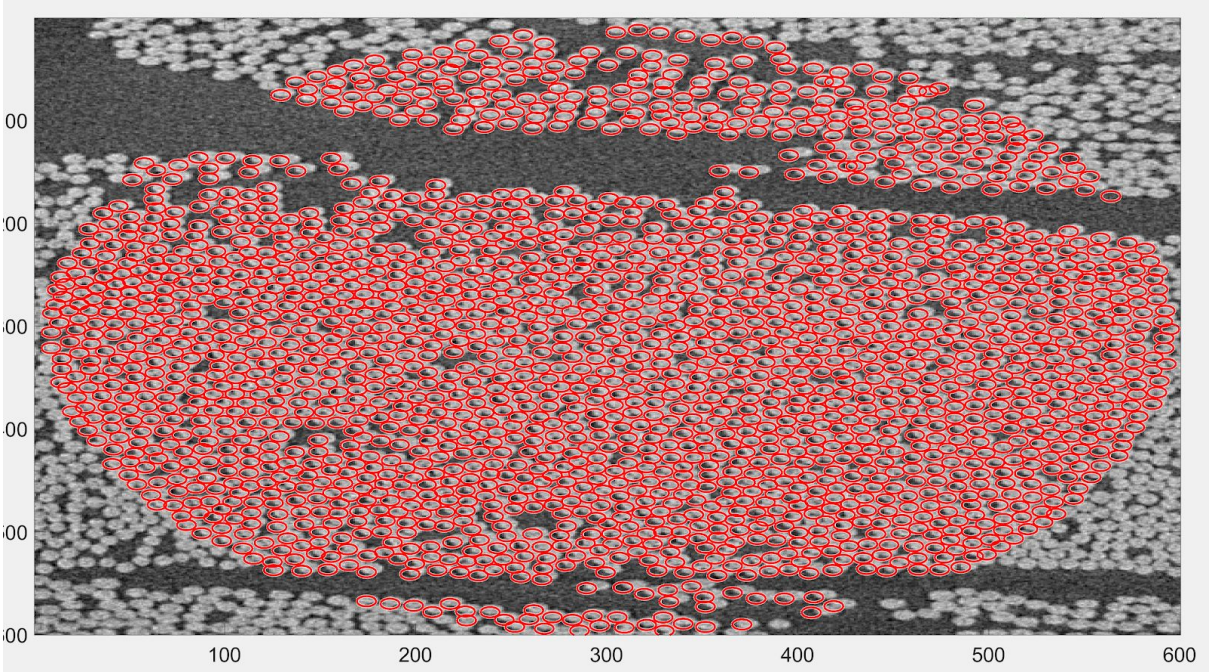


Figure 19: Homography transform between the high and medium resolution images. The transformation is seen to be translated slightly compared to the true result. This is also evident from inspection of **Figure 20** and **Figure 21**

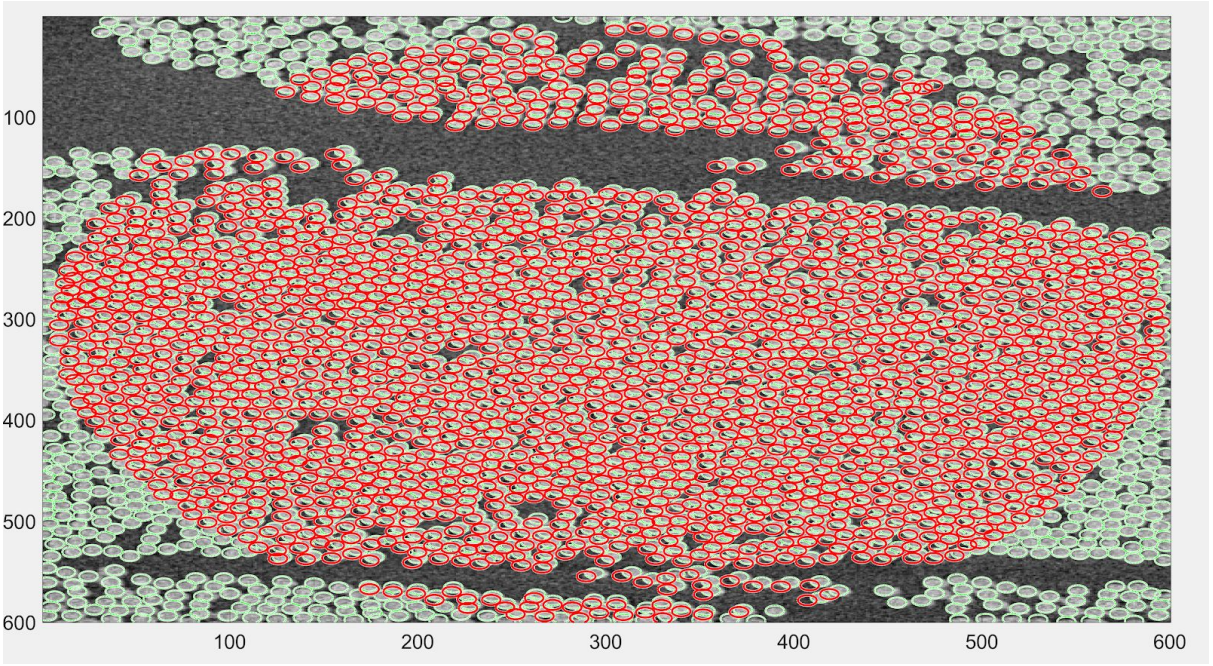


Figure 20: The found matches in green and red respectively. As mentioned in **Figure 19** a translation error is present. Below in **Figure 21** a zoomed version is available.

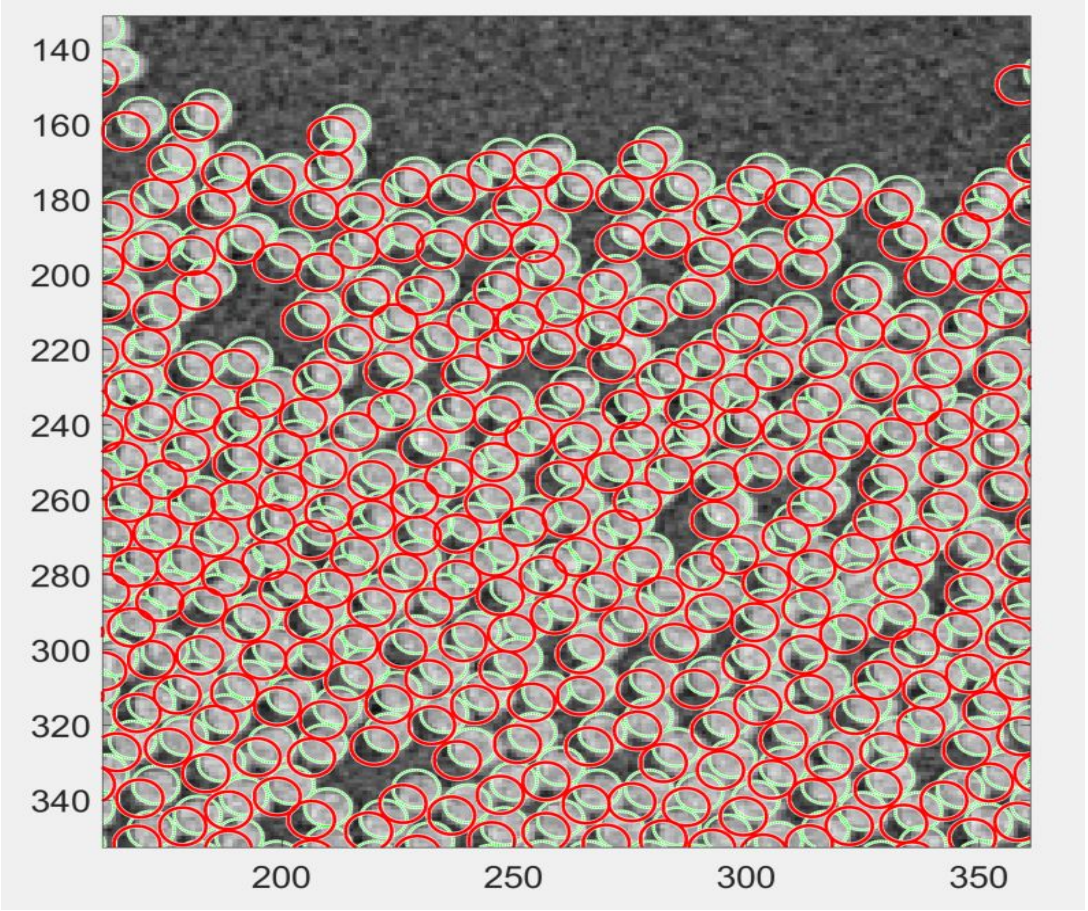


Figure 21: Close-up look at the translational error of the homography transformation. One reason why the match is not perfect might be because the slices are not taken from the same part of the volume.

Outlook and Additional Implementation:

- Try to experiment with the homography in order to attempt to obtain a better mapping. This would require refactoring of the code such that the pipeline is more smooth.
- Currently we use the Lowe threshold with Ransac to find valid matches. We have tried to use both our own implementation of the homography from a previous course as well as an online one. They yield similar results which indicates with some certainty that the problems are not to be found there.
- Another thing we could try is manually selecting 20 points for homography estimation.
- The error we found could stem from the fact that for the high resolution image have a lot of undefined zones around the black circle border. This gives a lot of empty space around the medium resolution image, i.e. we do not sample the whole image which reduces homography quality.
- Detect in scale space: We still need to detect points in one image, translate it to the other using the homography and then find the scale of the blops in the other image by taking the x-y coordinate and finding its equivalent max in scale-space. This is of course a general implementation which is redundant in this setting since all blops are reasonably the same size. (The exercise might make more sense if there were multi-sized objects in the image requiring the actual use of different scales).