

Figure 1. Segmented image obtained using (‘ECE565\_prob1.m’). Global thresholding algorithm applied to image (‘noisy\_fingerprint.tif’), resultant image used for binary segmented image.



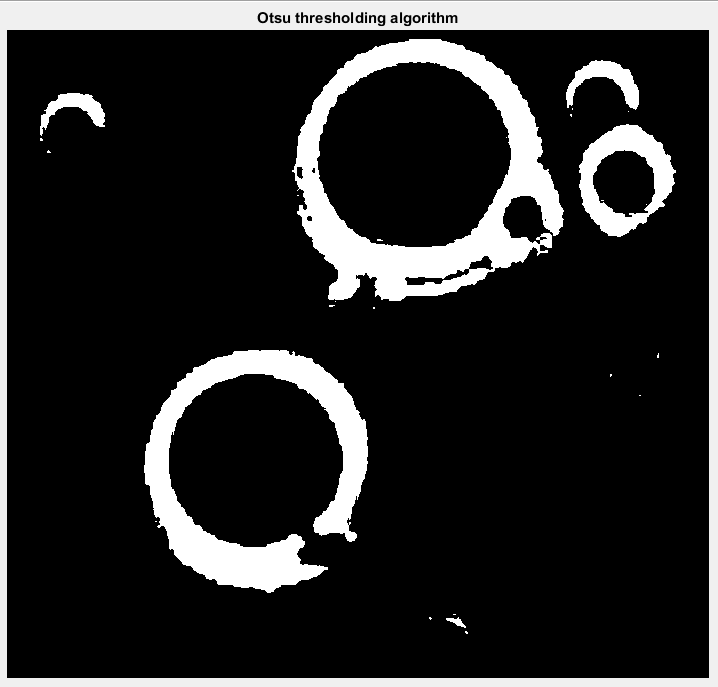


Figure 2a. Segmented image obtained using Otsu’s algorithm (‘ECE565\_prob2.m’) on image (‘polymersomes.tif’).



Figure 2b. Segmented image obtained using global thresholding algorithm (‘ECE565\_prob1.m’) on image (‘polymersomes.tif’)

Using the global thresholding algorithm from problem 1 on ‘polymersomes.tif’ shows a large imbalance in separating the foreground from the background. Otsu’s algorithm is able to better separate the foreground from the background. The resulting image from Otsu’s algorithm yields higher clarity by separating the membrane of the polymersome from the rest of the image. There are some parts of the image that are lost while using Otsu’s method, however, larger and more prominent high intensity values that pertain to the membrane are more easily visible. The global thresholding algorithm is not able to separate properly the high and low values causing significant lose in image clarity. The membrane is the key focus using these methods. Comparing the membranes between methods shows that the global thresholding algorithm from Problem 1 greatly extends the membrane size and smoothing this with empty space, causing a lose in image clarity and separation. Otsu’s method is able to separate the membrane from the space around and define its location without greatly affecting image clarity.



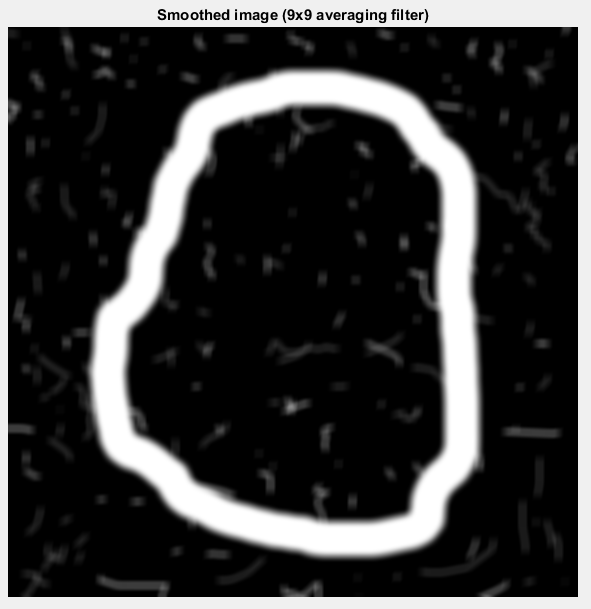
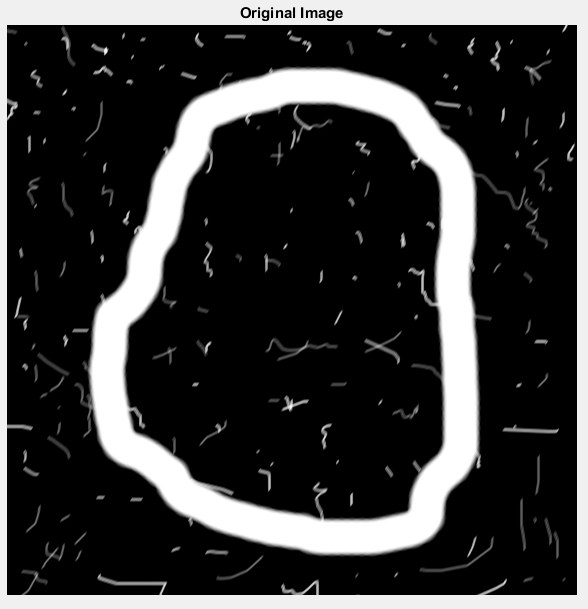


Figure 3a. Original image (left, ‘cicular\_stroke.tif’) and smoothed image (right, ‘circular\_stroke.tif’) using a 9x9 averaging filter ‘(ECE565\_prob3.m’)



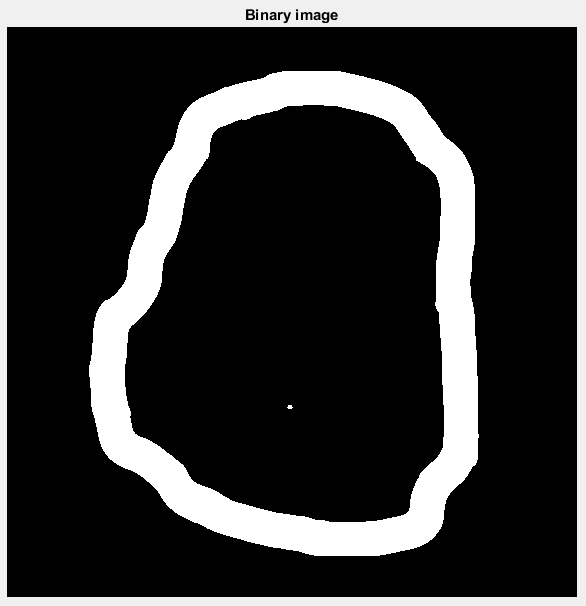


Figure 3b. Binary Image obtained from Figure3a, using smoothed image. Generated from function (‘ECE565\_prob3.m’)



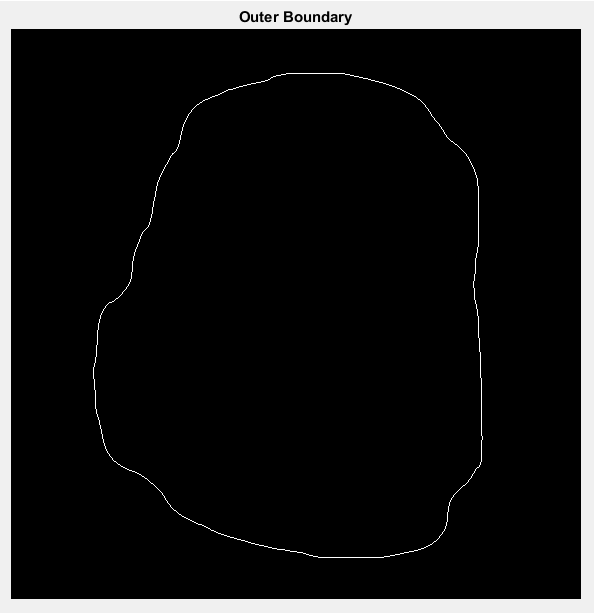


Figure 3c. Outer Boundary extracted from binary image obtained in Figure3b. (‘ECE565\_prob3.m’)

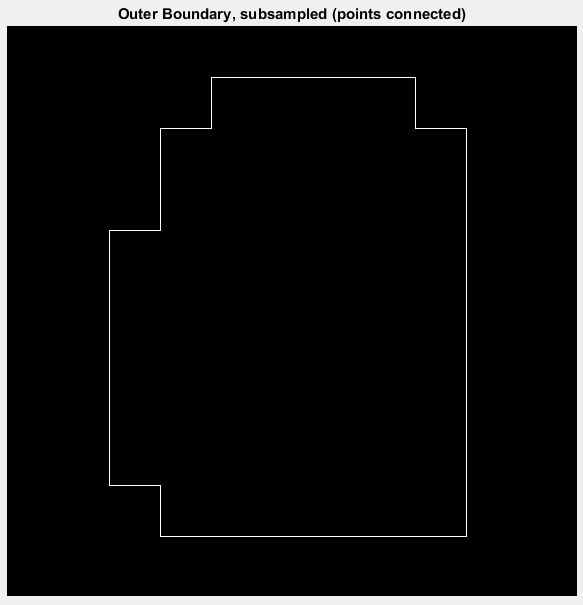
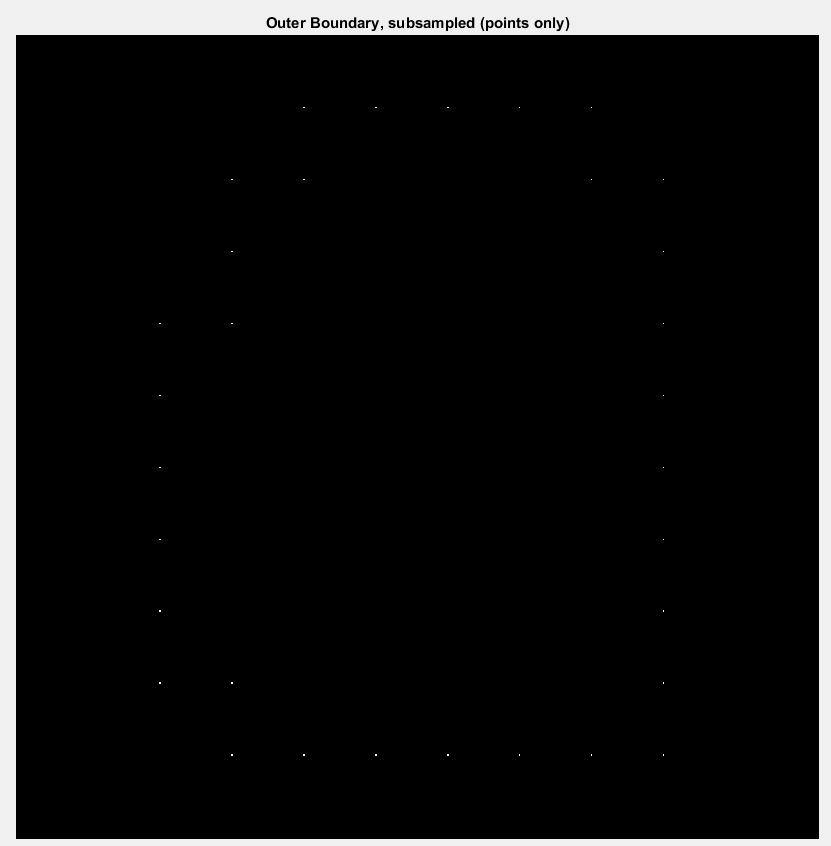


Figure 3d. Outer boundary subsampled, points only displayed (left) and points connected displayed (right).



Values obtained from ‘fchcode.m’, using subsampled points obtained (Figure 3d.) 8-connected chain and 4-connected chain parameters were both used to calculate chain code.

For 8-connected chain code:

Chain code:

c.fcc = 2220 2202 0000 6066 6666 6644 4444 2422

First Difference of code c.fcc:

c.diff = 0062 0626 0006 2600 0000 0600 0006 2600

Integer of minimum magnitude from c.fcc:

c.mm = 0000 6066 6666 6644 4444 2422 2220 2202

First difference of code c.mm:

c.diffmm = 0006 2600 0000 0600 0006 2600 0062 0626

Coordinates where the code starts

c.x0y0 = (8,3)

For 4-connected chain code:

Chain code:

c.fcc = 1110 1101 0000 3033 3333 3322 2222 1211

First Difference of code c.fcc:

c.diff = 0031 0313 0003 1300 0000 0300 0003 1300

Integer of minimum magnitude from c.fcc:

c.mm = 0000 3033 3333 3322 2222 1211 1110 1101

First difference of code c.mm:

c.diffmm = 0003 1300 0000 0300 0003 1300 0031 0313

Coordinates where the code starts

c.x0y0 = (8,3)



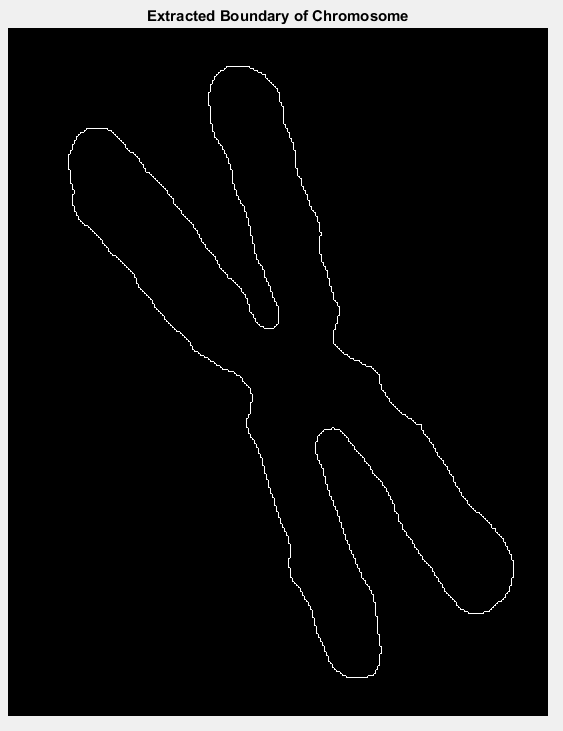
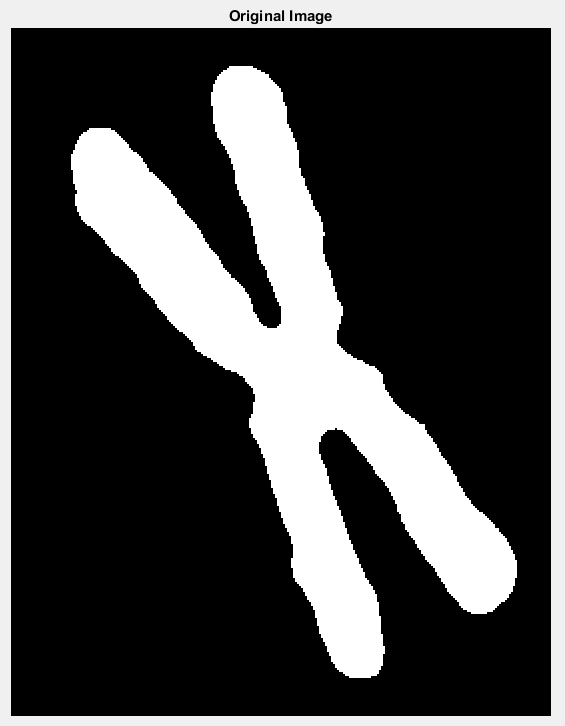


Figure 4a. Original image (left, ‘chromosome.tif’) and extracted boundary of image (right)



Fourier descriptors were obtained (‘ECE565\_prob4.m’) and output can be referenced in variable ‘z’. Total number of descriptors is 2688.



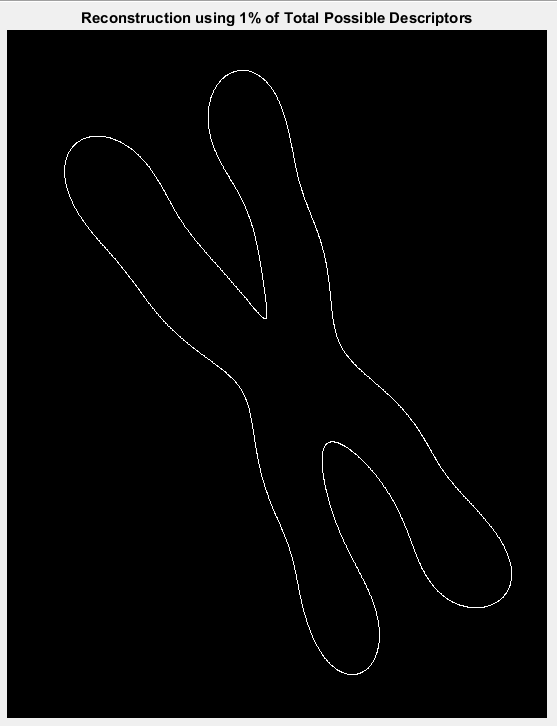
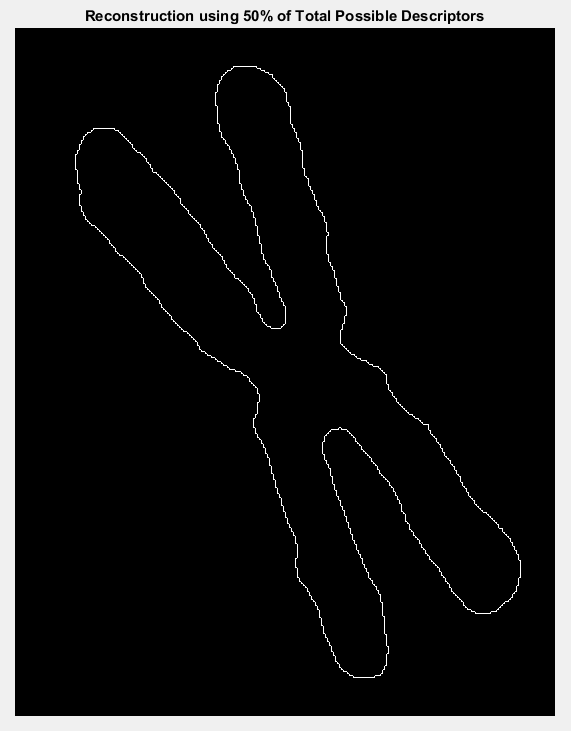


Figure 4c.

Reconstruction was performed on Fourier descriptors obtained in Problem 4, part b. Reconstruction was varied, using 50% of descriptor values from ‘z’ and 1% of descriptor values from ‘z’. Reconstruction performed with less descriptor values, 26 total, (Figure 4c. right) yield a smoother image. This is in comparison to using half the total descriptor values (Figure 4c left), 1344.