Advanced Programming in Engineering - Image Analysis

Christiaan Reurslag

s1495089

Exercise 1: Region labelling and ellipse fits

In this exercise an image (figure 1) showing a couple of cells was given. The first task was to separate the cells from the background using Utsu's method. To do this, the color image was first transformed to greyscale (figure 2) using my self-written function myRGB2Gray.

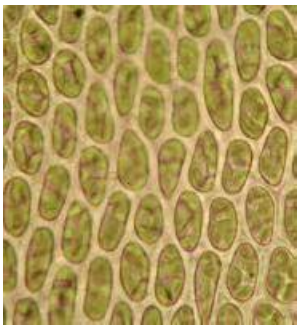

Fig 1: Original image

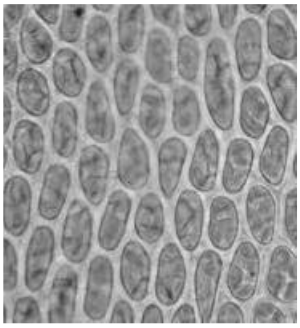

Fig 2: Image in greyscale

After that, a histogram (figure 3) of the image in greyscale was plotted using my own function myHistogram to check if we have indeed a bimodal distribution. From figure 3 follows we have indeed a bimodal distribution so we can use Otsu's method. The function myOTSU determines the optimum threshold value by minimizing the intra-class variance (figure 4) and after that the image is transformed to binary (figure 5).

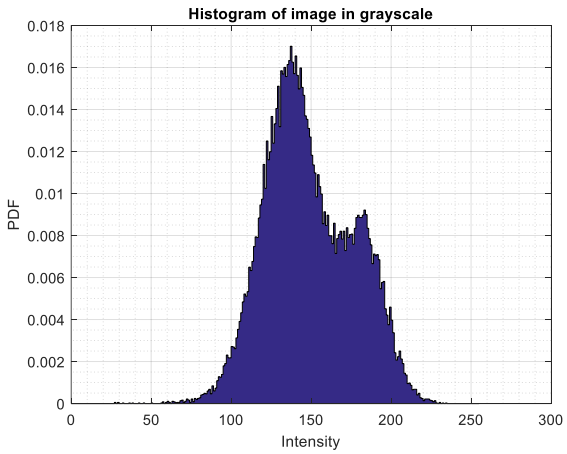

Fig 3: Histogram of image in grayScale

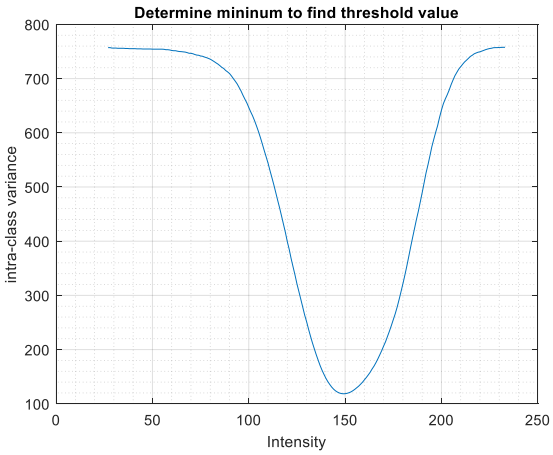

Fig 4: Intra-class variance (minimum at 149)

The number of cells is determined using a two pass algorithm. Every foreground element gets its own number (figure 6). Foreground elements smaller than 25 pixels are ignored. There are 61 cells visible in the image. For the second pass of the two pass algorithm, I found out that one iteration is not enough to give every foreground element its own single unique number. To solve this, the second pass is performed multiple times until there is no change anymore in the output matrix (see line 168 till 185 of APiE_1.m).

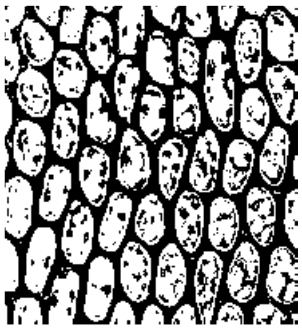

Fig 5: binary image

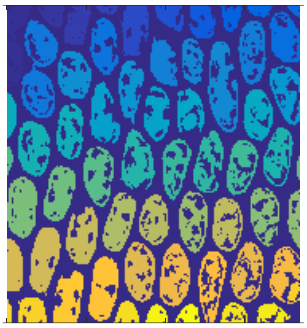

Fig 6: Every foreground element has its own number (color)

After that, also the area (number of pixels), center of mass, length of major and minor axes of fitted ellipse and eccentricity of fitted ellipse for every cell not touching the border is calculated (table 1). The original image with the fitted ellipses is shown in figure 7, also the center of masses are shown with a plus sign.

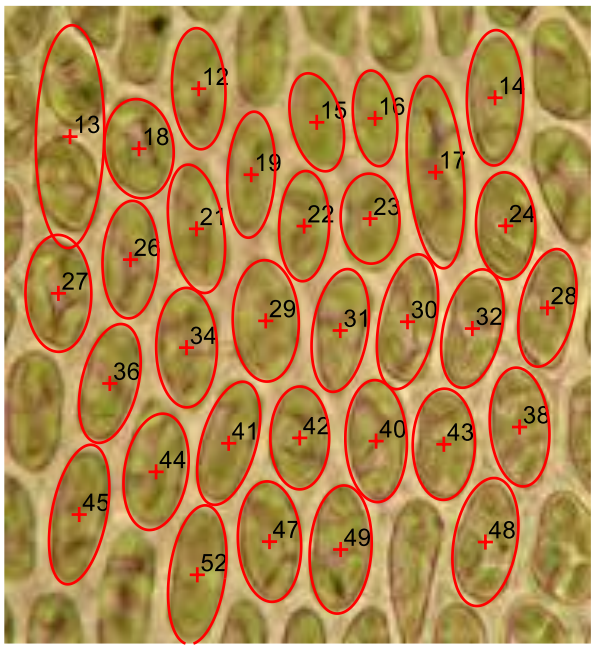

Fig 7: Image with fitted ellipses, center of masses and cell numbers

In figure 7 you can see that the fitted ellipses are quite accurate. Only in the left upper corner the algorithm made a mistake. Two separated cells are seen as only one cell. This is a result of the binary image. In the binary image the two cells are connected with a thin line of only one pixel wide so the two cells are assigned the same number by the two pass algorithm.

Cell number (fig. 7)	Area (number of pixels)	Center of mass (hor.)	Center of mass (vert.)	Major axis	Minor axis	Eccentricity
12	396	61.5	26.5	90.2	17.5	0.898
13	811	21.1	41.5	305.4	28.7	0.952
14	444	154.4	29.3	112.6	19.5	0.909
15	367	98.5	37.0	61.5	15.9	0.861
16	256	116.8	35.8	56.8	11.6	0.892
17	632	135.8	52.7	228.6	18.9	0.958
18	430	42.7	45.3	60.9	29.2	0.721
19	360	77.9	53.5	97.9	13.9	0.926
21	420	60.8	70.4	103.3	18.1	0.908
22	338	94.5	69.6	74.7	15.4	0.891
23	313	115.2	67.2	49.8	21.3	0.756
24	417	157.8	69.5	70.4	21.6	0.832
26	387	40.1	80.1	85.0	19.1	0.881
27	469	17.5	90.7	84.0	26.6	0.826
28	394	170.9	95.2	87.7	18.1	0.891
29	459	82.5	99.3	90.6	26.8	0.839
30	384	127.0	99.6	115.2	19.5	0.912
31	360	105.9	102.3	94.2	18.6	0.896
32	381	147.3	101.7	89.4	20.7	0.877
34	434	57.7	107.7	87.7	22.7	0.861
36	454	33.6	118.9	90.7	19.6	0.886
38	385	162.1	132.6	87.1	21.5	0.868
40	385	117.1	137.0	91.4	23.2	0.864
41	468	70.9	137.7	99.1	18.4	0.902
42	388	93.2	136.0	65.0	21.3	0.820
43	422	138.4	138.0	75.9	23.8	0.829
44	471	48.1	146.6	84.7	24.7	0.842
45	556	24.0	160.0	123.5	18.5	0.922
47	485	83.7	168.5	89.3	24.4	0.852
48	425	151.4	168.7	103.1	24.4	0.874
49	500	106.0	171.0	101.2	23.3	0.877
52	542	61.0	178.9	120.4	19.2	0.917

Tab 1: Numerical values

For this assignment I only made use of very basic MATLAB functions. The more complex algorithms were self-written. This assignment can be finished much sooner when build-in MATLAB functions were used. The performance of the script can also be improved when build-in MATLAB functions were used because these MATLAB functions are precompiled. Performance can also be improved when intensities are stored as 8 bit unsigned integers instead of doubles, but the accuracy will be less in this case due to rounding errors.

Exercise 2: Cross-correlation and displacement

In this exercise two images (figure 8 and figure 9) were given. In this exercise the cross-correlation between figure 8 and itself is calculated (figure 10). The same is done for figure 9 (figure 11).

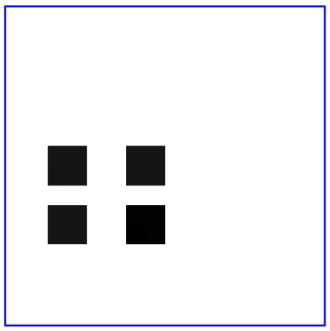

Fig 8: pattern1.tif

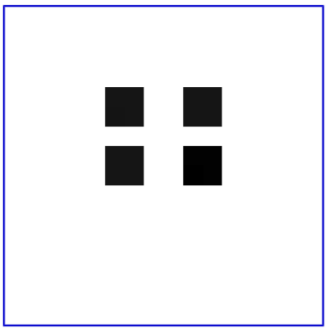

Fig 9: pattern2.tif

From figure 10 and figure 11 follows the maximum of the cross-correlation is exactly in the middle. This is expected because when an image on top of the same image is not shifted we have of course the closest match. When the mask is moved 4 pixels to the left, right, up or down direction we see again a peak in the cross-correlation. This is because 2 of the 4 black squares overlap again. When the mask is moved 4 pixels to the left or right and 4 pixels up or down there is a smaller peak visible in the cross-correlation because this time only one black square overlap.

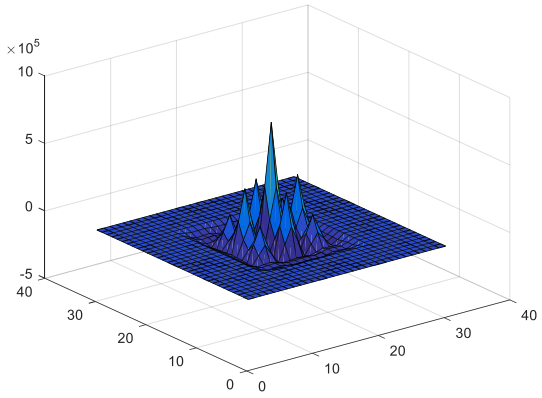

Fig 10: Cross-correlation of figure 8 and itself

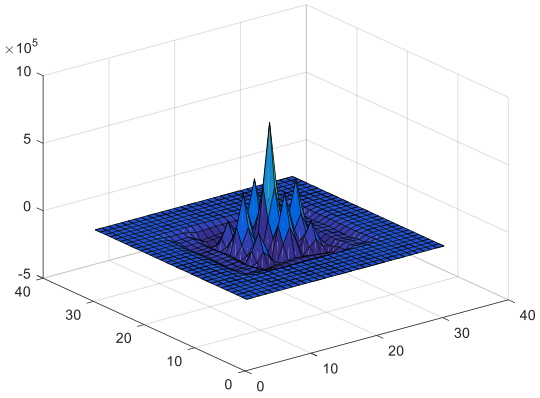

Fig 11: Cross-correlation of figure 9 and itself

In figure 12 the cross-correlation between figure 8 and figure 9 is calculated.

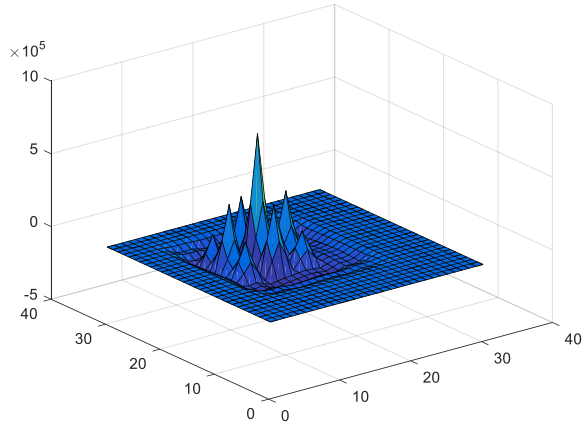

Fig 11: Cross-correlation between figure 8 and figure 9

We see again the same pattern, but this time the pattern is shifted. This is because figure 8 and figure 9 are the same image except for a shift. From the cross-correlation the displacement is calculated and found to be (-3,3). From this follows that after a time dt, the 'particles' in figure 8 have moved -3 pixels downward (positive direction is downwards) and 3 pixels to the right.

Exercise 3: Particle Image Velocimetry

In this exercise again two images were given (figure 12 and 13). The first task was to eliminate the effects of inhomogeneous lighting by applying adaptive histogram equalization. The algorithm make use of a sliding window approach. This method gives the best result but is more computational expensive. The result is shown figure 14 and figure 15.

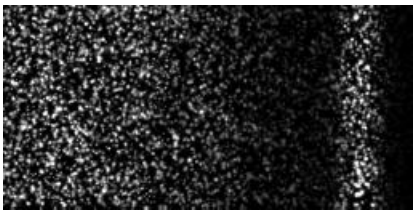

Fig 12: Original image (E001_1.tif)

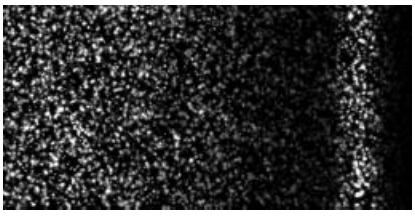

Fig 13: Original image (E001_2.tif)

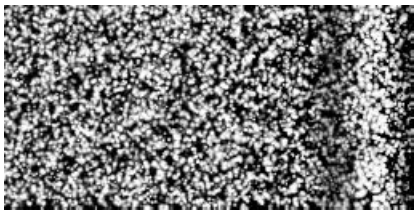

Fig 14: Figure 12 after adaptive histogram equalization

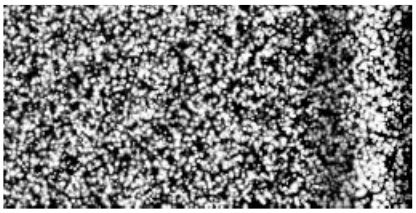

Fig 15: Figure 13 after adaptive histogram equalization

The cumulative distribution function (CDF) of both figures before and after adaptive histogram equalization are shown in figure 16 and figure 17.

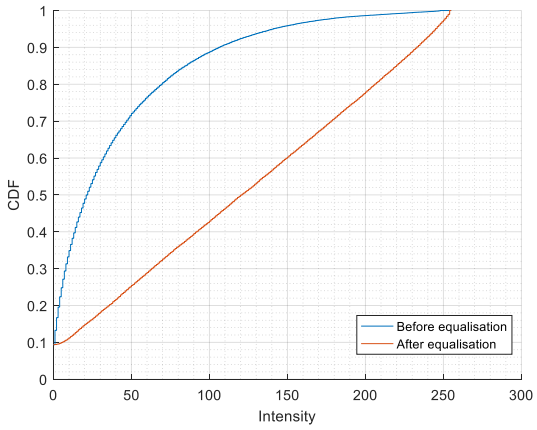

Fig 16: CDF before and after equalization of figure 12

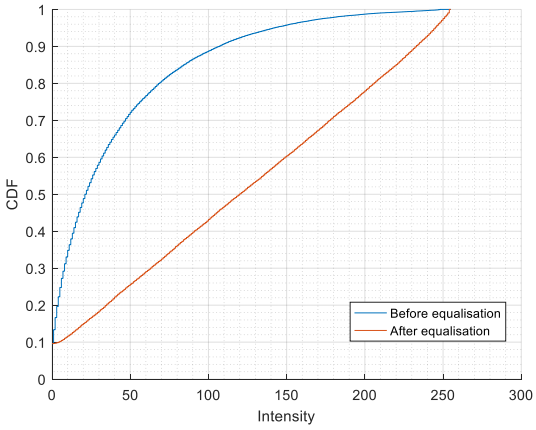

Fig 17: CDF before and after equalization of figure 13

From figure 16 and 17 we can see the adaptive histogram equalization has worked. After the equalization the CDF is linear. Also the original images look better after the equalization. The individual particles are better visible and the effects of inhomogeneous lighting are reduced.

The cross-correlation function from exercise two is used to calculate the velocity field. The result is shown in figure 18. The color shows the size of the velocity (speed) and the red arrows the direction.

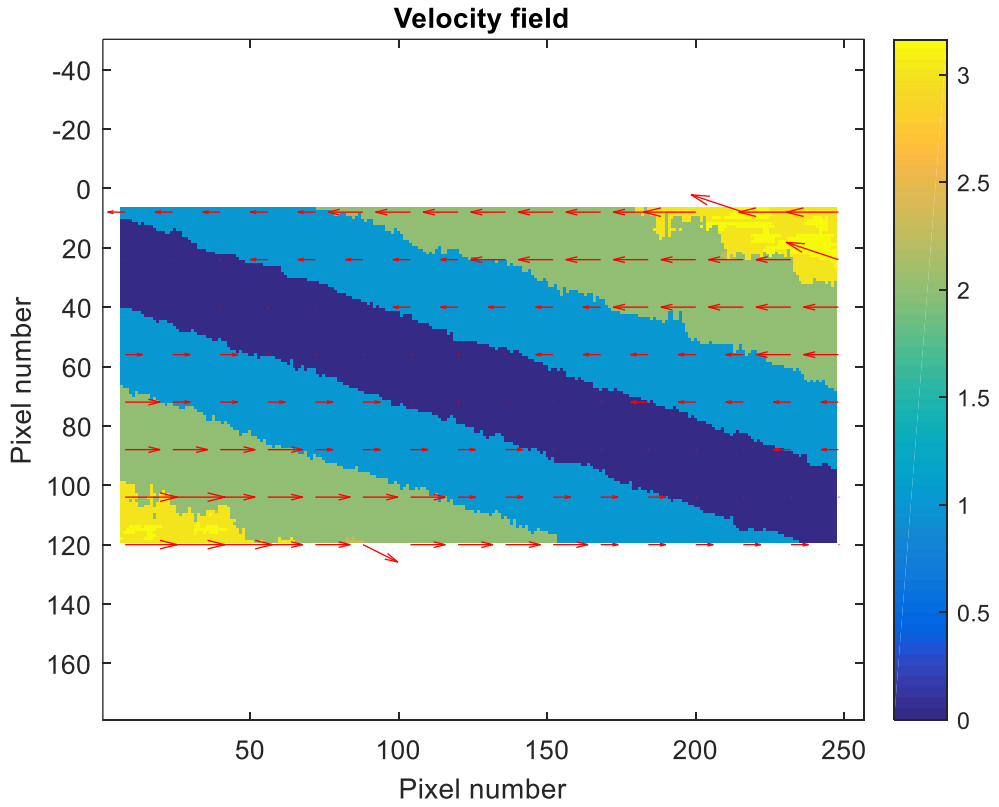

Fig 18: Velocity field

It takes quite a lot of time to execute the script (around 11 minutes). The performance can be improved by using the tilling approach for the adaptive histogram equalization. Performance can also be improved when intensities are stored as 8 bit unsigned integers instead of doubles, but the accuracy will be less in this case due to rounding errors. Making use of the MATLAB build-in functions will also speed up the execution.