Medical Image Analysis Assignment

Table of Contents

BIOM9027 Ludwig Tranheden z5129849

1)

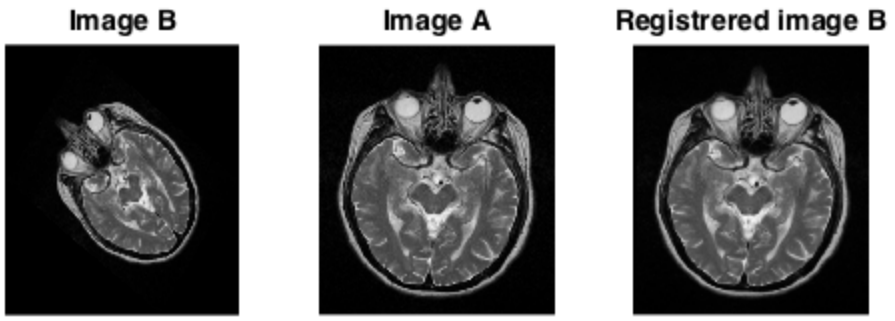
```
clear all
close all
clc
clf

load MyImages

%a
RGB_im = ind2rgb(Question1Image,Question1cmap); %Generete RGB-image.

%b
Gray_im = rgb2gray(RGB_im); %Convert to gray scale.

%c
seg = Gray_im == 1; %Segment the allocated number

%d
subplot(1,3,1)
imshow(RGB_im);
title('1a - RGB','Fontsize',14)

subplot(1,3,2)
imshow(Gray_im);
title('1b - Gray','Fontsize',14)

subplot(1,3,3)
imshow(seg);
title('1c - Binary','Fontsize',14)
```

2)

```
clc
clf
close all

%a

%[im3,TFORM,overlay] =
 Registrationdemo(Question2Image_B,Question2Image_A); %Select common
 points and

      %generate the affine transfrom object.
%Did the above and saved the data in MyImages.

subplot(1,3,1)
imshow(Question2Image_B);
title('Image B','Fontsize',14)

subplot(1,3,2)
imshow(Question2Image_A);
title('Image A','Fontsize',14)

subplot(1,3,3)
imshow(im3); %The registrered image B.
```

```
title('Registrered image B','Fontsize',14)

%b
A=TFORM.tdata.T(1:2,1:2); %The affine transformation matrix without
 translation data.
[U,S,V]=svd(A); %Break down into three components: rotation U,
 deformation S (diagonal -> scaling), rotation V.
xscale = S(1,1); %Scaling value along x-axis.
yscale = S(2,2); %Scaling value along y-axis.
display('The amount that imageB has been enlarged in x direction after
 the first rotation')
display(xscale)
display('The amount that imageB has been enlarged in y direction after
 the first rotation')
display(yscale)

%c
ROTX = [1 0]*A; %How much imageB was rotated (Just multiply affine
 transform by x base vector).
x_rotation = atan2(ROTX(2),ROTX(1))*360/(2*pi); %The angle in degrees.
display('How much imageB was rotated in degrees')
display(x_rotation)
```

```
The amount that imageB has been enlarged in x direction after the
 first rotation

xscale =

    1.0388

The amount that imageB has been enlarged in y direction after the
 first rotation

yscale =

    0.8350

How much imageB was rotated in degrees

x_rotation =

   48.8640
```

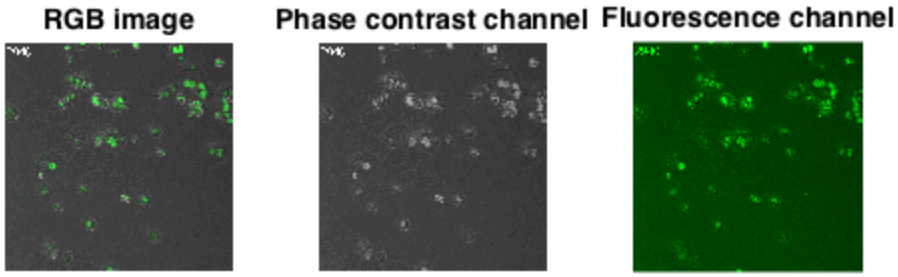


3)

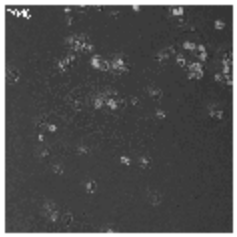
```
clc
clf
close all

warning('off','all') %Supress "Image is too big to fit on screen;
 displaying at 67%".

%a
Q3_RGB = ind2rgb(Question3Image,Question3cmap); % Convert to RGB.

%b
figure(1)
subplot(1,3,1)
imshow(Q3_RGB);
title('RGB image','Fontsize',14)

subplot(1,3,2)
Q3_gray = rgb2gray(Q3_RGB); %Phase contrast channel.
imshow(Q3_gray)
title('Phase contrast channel','Fontsize',14)


subplot(1,3,3)
```

```
Q3_g = Q3_RGB;
Q3_g(:,:,[1, 3]) = 0; %Fluorescence channel.
imshow(Q3_g)
title('Fluorescence channel','Fontsize',14)

%c
Segmentnuclei_self = Q3_g(:,:,2) > 0.5; %Manually set threshold.
threshold=graythresh(Q3_g(:,:,2));  %Global threshold through Otsu's
 method.
Segmentnuclei_otsu=im2bw(Q3_g(:,:,2),threshold); %Create binary image
 from the threshold.

figure(2)
subplot(1,3,1)
imshow(Q3_gray)
title('Grayscale image','Fontsize',14)

subplot(1,3,2)
imshow(Segmentnuclei_self)
title('Manual threshold','Fontsize',14)

subplot(1,3,3)
imshow(Segmentnuclei_otsu)
title('Otsu method','Fontsize',14)

%Otsu`s method contains more of the nuclei then the manually set.
 Didn`t
%spend much time on the manually set threshold though. If carefully
 setting
%the threshold by some trial and error it would probably be better
 then
%Otsu´s. I.e manual will be better then automated tresholds if you
 have
%the time to tune it.

%d
figure(3)
[BW,threshold] = edge(Q3_gray,'log'); %Find edges using LOG edge
 detection method.
subplot(2,2,1)
imshow(BW)
title('After LOG-filter','Fontsize',14)

BW2 = imclose(BW,strel('disk',6)); %Close to smooth and connect
 irregular features
subplot(2,2,2)
imshow(BW2)
title('After closing','Fontsize',14)

BW2 = imfill(BW2,'holes'); %Fill the holes in the cell bodies.
subplot(2,2,3)
imshow(BW2)
title('After filling holes','Fontsize',14)
```

```
BW2 = imopen(BW2,strel('disk',10)); %Open to remove small scatter
                                    %while preserving the shape and
 size of the cell bodies.
subplot(2,2,4)
imshow(BW2)
title('After opening, done','Fontsize',14)

figure;
imshow(Q3_RGB)
hold on
himage = imshow(BW2);
himage.AlphaData = 0.4; %Add transparency.
title('The segmenting on top of the original image','Fontsize',14)
```



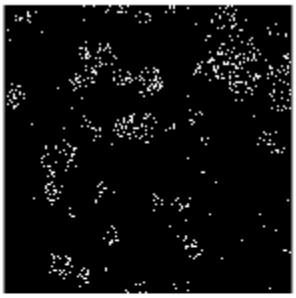
Grayscale image



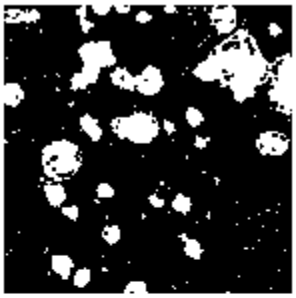
Manual threshold

Otsu method

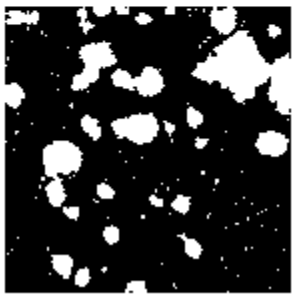
After LOG-filter



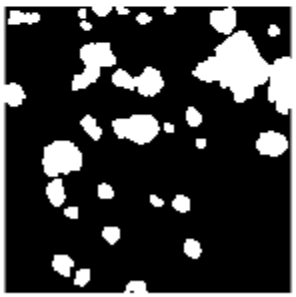
After closing

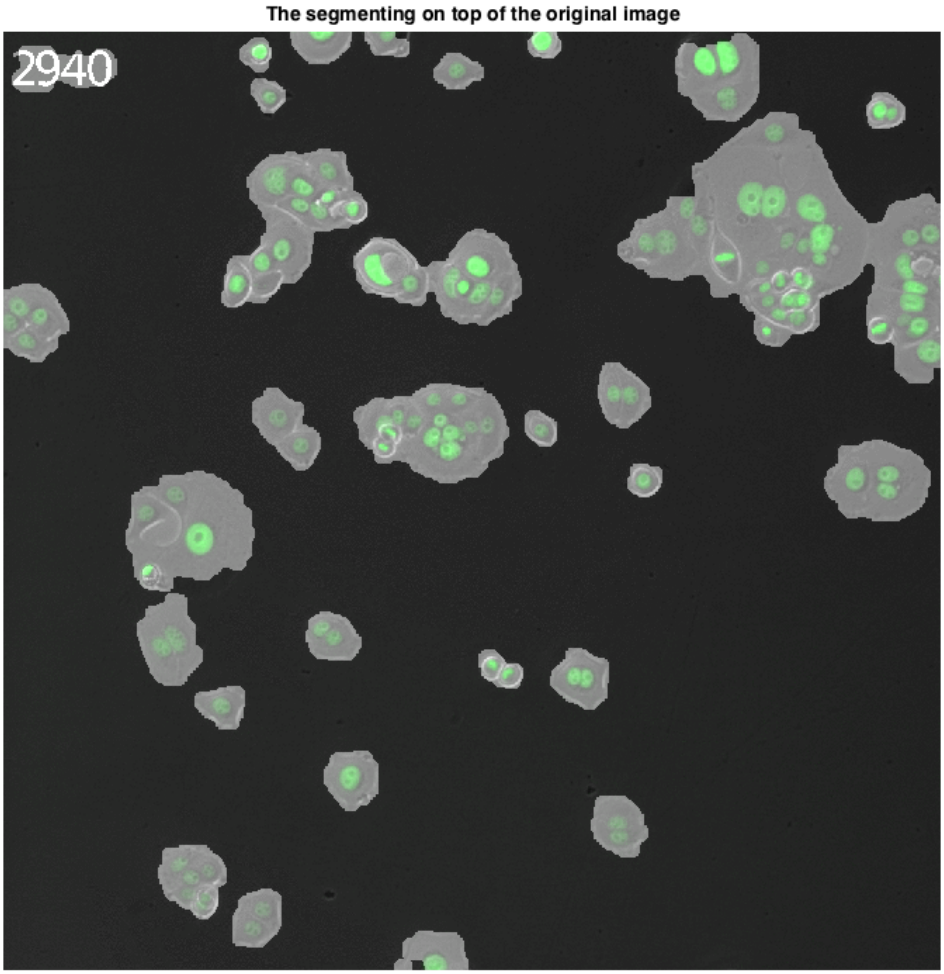


After filling holes



After opening, done



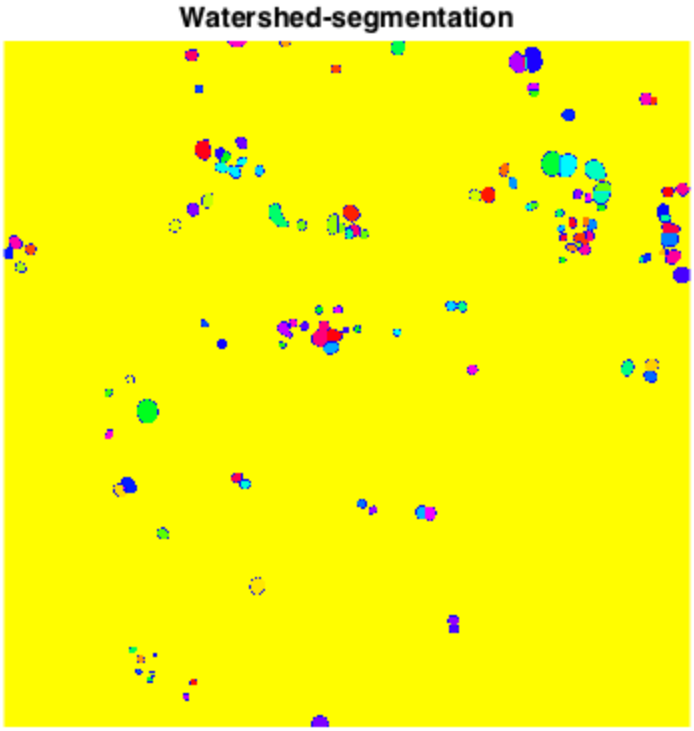


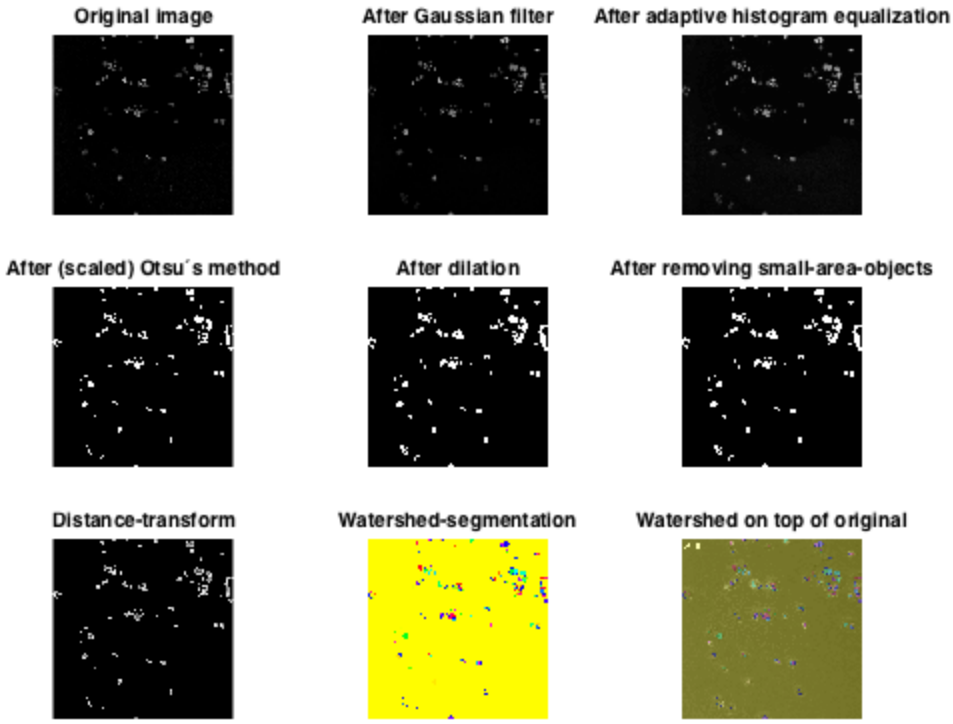
4)

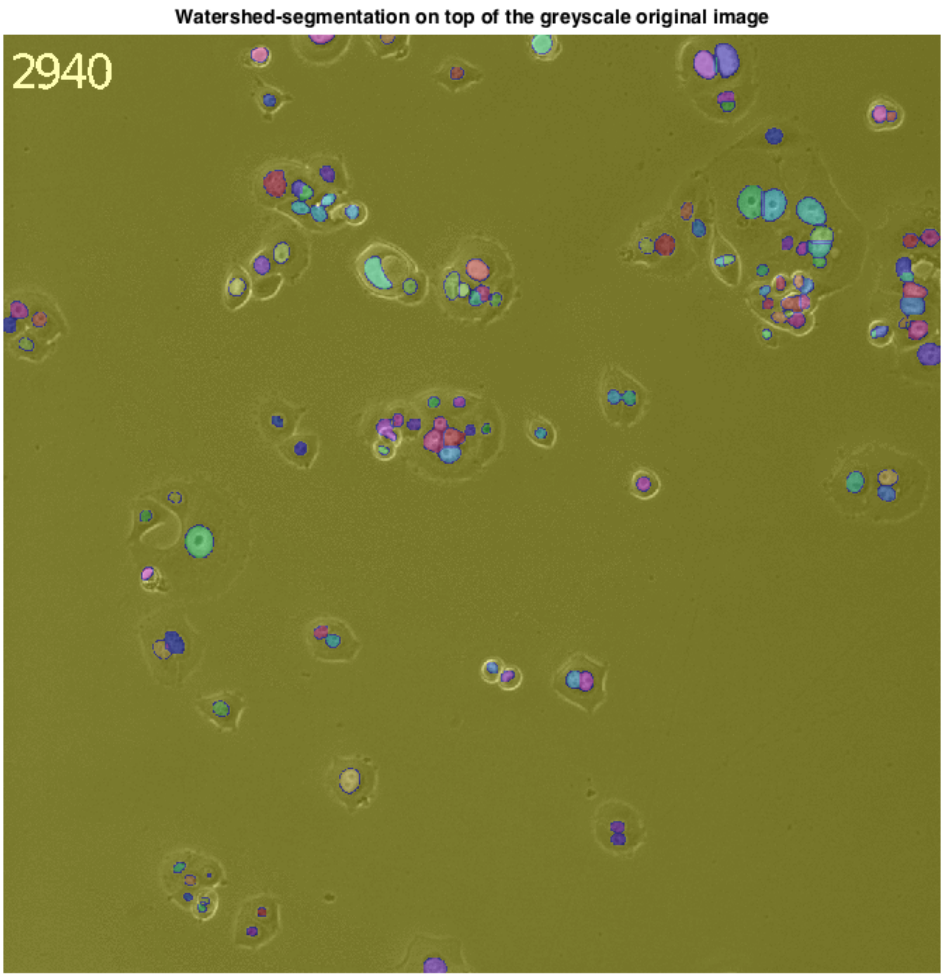
```
close all
clc
clf

img = Q3_RGB(:,:,2) - Q3_RGB(:,:,1); %Remove the cell bodies but not
 nuclei.

subplot(3,3,1)
imshow(img)
title('Original image','Fontsize',10)


img = imgaussfilt(img,1.1); %Gaussian filter for noise removal with
 edge preservation.
subplot(3,3,2)
imshow(img)
title('After Gaussian filter','Fontsize',10)
```

```
img = adapthisteq(img); %Imrove local contrast and enhance the
 definitions of edges
                        %in each region through adaptive histgram
 equalization.
subplot(3,3,3)
imshow(img)
title('After adaptive histogram equalization','Fontsize',10)


threshold=graythresh(img); %Create binary image through Otsu's method.
img=im2bw(img,threshold*0.8); %Scale threshold (absence of noise).
subplot(3,3,4)
imshow(img)
title('After (scaled) Otsu´s method','Fontsize',10)


img = imdilate(img, strel('disk',2)); %Dilate to enlarge nuclei.
subplot(3,3,5)
imshow(img)
title('After dilation','Fontsize',10)

img = bwareaopen(img, 75); %Remove scatter.
subplot(3,3,6)
imshow(img)
title('After removing small-area-objects','Fontsize',10)


img = imopen(img, strel('disk',5)); %Open to remove small objects but
 preserve nuclei.
subplot(3,3,7)
imshow(img)
title('After opening','Fontsize',10)


C = ~img; %Invert binary image.
D = -bwdist(C,'quasi-euclidean'); %Distance transform using quasi-
euclidean distance.

D(C) = -inf; %Background to -inf.
regmax = imregionalmax(D); %Find regional maxima.
regmax = imdilate(regmax,strel('rectangle',[1 1]));  %Dilate to
 coalesce several regional maximas of nuclei.
regmax = imdilate(regmax,strel('disk',1));
D(regmax) = -inf; %To -inf.
% Watershed will now fill from the centre of nuclei to its edge.

subplot(3,3,7)
imshow(D,[],'InitialMagnification','fit');
title('Distance-transform','Fontsize',10)

L = watershed(D); %Do the watershed-transform.
rgb = label2rgb(L,'hsv','b', 'shuffle');
subplot(3,3,8)
```

```
imshow(rgb,'InitialMagnification','fit')
title('Watershed-segmentation','Fontsize',10)


figure;
imshow(rgb,'InitialMagnification','fit') %Show in seperate image for
 visibility.
title('Watershed-segmentation','Fontsize',14)

figure(1)
subplot(3,3,9)
imshow(Q3_gray)
hold on
himage = imshow(rgb); %Show segmentation on top of original greyscale
 image.
himage.AlphaData = 0.3;
title('Watershed on top of original','Fontsize',10)


figure;
imshow(Q3_gray)
hold on
himage = imshow(rgb); %Show in seperate image for visibility.
title('Watershed-segmentation on top of the greyscale original
 image','Fontsize',14)
himage.AlphaData = 0.3;
```




Original image
After Gaussian filter
After adaptive histogram equalization
After (scaled) Otsu´s method
After dilation
After removing small-area-objects
Distance-transform
Watershed-segmentation
Watershed on top of original



Published with MATLAB® R2015a