Lab 5:
Segmentation of cells in microscopy images

Maria Magnusson, 2007-2021,
Computer Vision Laboratory, Department of Electrical Engineering,
Linköping University, Sweden

1 Introduction

Read all instructions before the lab-exercise and repeat necessary theory from the lectures and the book. Problems marked with a pointing hand are supposed to be solved as a preparation before the lab-exercise.
The lab is inspired by the article:
Segmentation of Cytoplasms of cultured cell by Amin Allalou et al.
It is good to browse the article so that you understand it approximately.

A computer symbol indicates that a MATLAB-script has to be written and demonstrated during the lab-assignment.

Start by extracting the zip-archive containing the files `c9minpeps2.bmp`, `SegmentCellsPrel.m` and `Overlay.m` into a suitable folder, e.g. Lab5.

2 Introductory watershed experiments

At first we will try the watershed algorithm on an inherent image in MATLAB. Create a file `MyWatershed.m` with the subsequent contents and execute it.

```
% Read a color image and convert to a gray-scale image
% ====================================================
[X,map] = imread('corn.tif');
newmap = rgb2gray(map);
corn = ind2gray(X,newmap);
im1gray = double(corn);
figure(1), imagesc(im1gray,[0 255]), colormap(gray), colorbar,
title('original image');

% Negate the image
% ================
im2gray = double(255-im1gray);
figure(2), imagesc(im2gray,[0 255]), colormap(gray), colorbar,
title('negated image = landscape');

% Prepare for smoothing
% =====================
kernel = [1 2 1; 2 4 2; 1 2 1] / 16;
D = im2gray;
k=1;
while k<1
  D = conv2(D, kernel, 'same');
```

```
  k = k+1;
end
figure(3), imagesc(D);
colormap(gray), colorbar;
title('smoothed landscape');

% Perform watershed transform
% ===========================
W1 = double(watershed(D));
figure(4), imagesc(W1);
colormap(colorcube(300)), colorbar;
title('Watershed transform of smoothed landscape')
```

QUESTION 1: Why is it necessary to negate the original image?

Because the watershed algorithm operates such that it finds areas encapsulated by white ridges

QUESTION 2: Look at the resulting watershed image W1. Oversegmentation is evident. Why does oversegmentation happen?

Because it still has a large variety within the kernel areas => too many local minima

QUESTION 3: Oversegmentation can be prevented by smoothing the image. Adjust the while-loop so that most of the individual corn grains in the resulting image become segmented properly. How many times did you need to convolve with the given kernel?

Around 5 times

QUESTION 4: See the matrix below.

```
A =
 10 10 10 10 10 10 10
 10  7  7 10 10 10 10
 10  7  7 10 10 10 10
 10 10 10 10 10 10 10
 10 10 10 10  2  2 10
 10 10 10 10  2  2 10
 10 10 10 10  2  2 10
 10 10 10 10 10 10 10
```

What is the result when using `B=imhmin(A,4)` ? Also explain the look of B.

ans =

```
   10   10   10   10   10   10   10
   10   10   10   10   10   10   10
   10   10   10   10   10   10   10
   10   10   10   10   10   10   10
   10   10   10   10    6    6   10
   10   10   10   10    6    6   10
   10   10   10   10    6    6   10
   10   10   10   10   10   10   10
```

we raise the waterlevel by 4
so the 7:s overflow and 2:s rise to 6:s

An alternative to smoothing is to use the function `imhmin.m` Add the following code to `MyWatershed.m`.

```
% Prepare for imhmin treatment
% ============================
E = im2gray;
E = imhmin(E,0);
figure(5), imagesc(E);
```

```
colormap(gray), colorbar;
title('imhmin treated landscape');

% Search local min
% ================
Emin = imregionalmin(E);
CC = bwconncomp(Emin,8);
ImLabel = labelmatrix(CC);
figure(6), imagesc(ImLabel);
colormap(colorcube(300)), colorbar;
title('local means (water holes)')

% Perform watershed transform
% ===========================
W2 = double(watershed(E));
figure(7), imagesc(W2);
colormap(colorcube(300)), colorbar;
title('Watershed transform of imhmin treated landscape')
```

QUESTION 5: Look at the resulting watershed image W2. Oversegmentation is evident. Adjust the parameter in `imhmin` to get rid of most of the oversegmentation. Write your `imhmin` command below. Also mention what happens with the local means (water holes).

E = imhmin(E,30);
the contours are better preserved

QUESTION 6: Compare the two watershed results, W1 and W2. Which one delineates the individual corn grains best?

W2 is a bit better

3 Segmentation of cells, cytoplasm and padlock-signals

See the microscopy image in Figure 1. The cell kernels are blue. There are also so called padlock-signals of two different types, red and green. The cytoplasm (invisible) is located around the cell kernels. The hypothesis is that a pixel rather close to, but outside, the cell kernels contains cytoplasm and belongs to the closest cell kernel.
The task is to determine which padlock signals belong to the respective cell. The desired result is shown in Figure 2, where the cytoplasm for each cell has been delineated (in yellow). For one cell, the corresponding green padlock signals have been detected and marked with circles (in cyan).
The first part of the MATLAB script `SegmentCellsPrel.m` is shown below. A color microscopy image is read and shown in figure(1). Its three color components are shown in figure(2), figure(3) and figure(4). The blue component contains the cells kernels and are treated first. A histogram of the central part of the image is calculated and shown in figure(5). The reason to use the central part is to lower the influence of the dark background and the diffuse magenta-colored objects below-left in Figure 1.

Execute `SegmentCellsPrel.m` in MATLAB and verify the result in figure(1)-figure(5)!

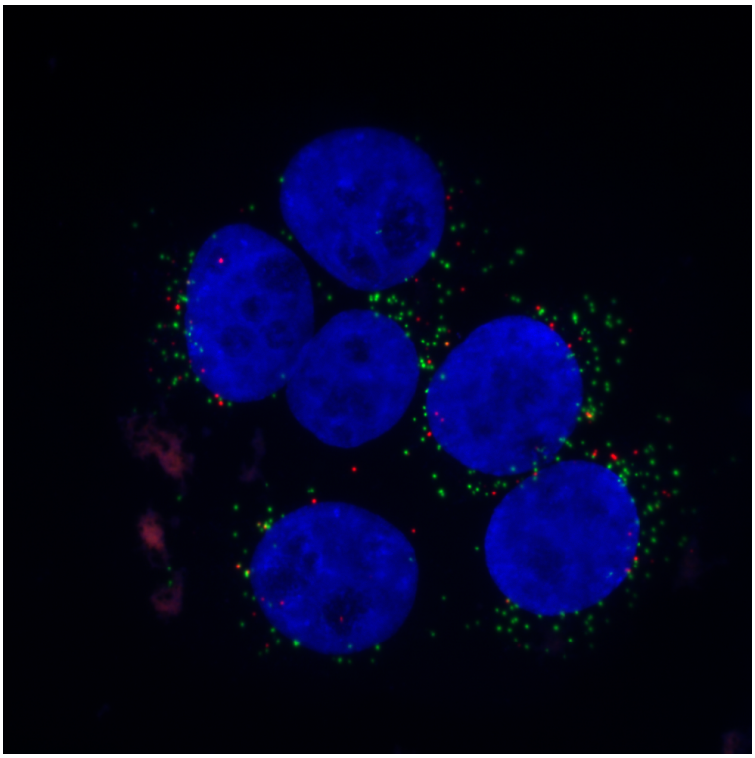

Figure 1: Original image.

```
% ################################
% Cell segmentation, preliminary
% ################################

% Read a color image
% ==================
im1 = double(imread('C9minpeps2.bmp'));
figure(1), imshow(im1/255);
title('Original color image');

% Look at the three colour components RGB
% =======================================
im1r=im1(:,:,1); im1g=im1(:,:,2); im1b=im1(:,:,3);
im1bSmall = im1(200:800,200:800,3);
figure(2), imagesc(im1r,[0 255]), axis image, colormap(gray), colorbar;
title('Red image');
figure(3), imagesc(im1g,[0 255]), axis image, colormap(gray), colorbar;
title('Green image');
figure(4), imagesc(im1b,[0 255]), axis image, colormap(gray), colorbar;
title('Blue image');

% Compute histogram of central part of the blue image
% ===================================================
histo = hist(im1bSmall(:),[0:255]);
figure(5), stem(histo);
```

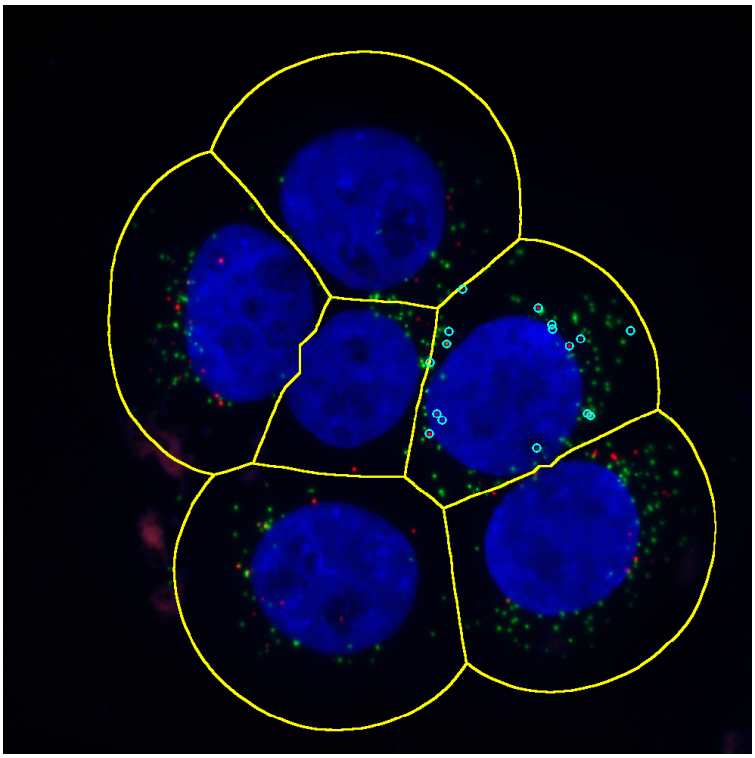

Figure 2: Desired result image.

3.1 Thresholding and Morphological image processing

The second part of the MATLAB script `SegmentCellsPrel.m` is shown below. The image is manually thresholded at 42 and shown in figure(6). Then the image is morphologically processed and shown in figure(7).

```
% Perform thresholding
% ====================
im1bT = im1b > 42;
figure(6), imagesc(im1bT), axis image, colormap(gray), colorbar;
title('After thresholding');

% Perform opening
% ===============
SE4 = [0 1 0;
       1 1 1;
       0 1 0];
SE8 = [1 1 1;
       1 1 1;
       1 1 1];
tmp = im1bT;
tmp = imerode(tmp, SE4);
tmp = imdilate(tmp, SE4);
im1bTmorph  = tmp;
figure(7), imagesc(im1bTmorph), axis image, colormap(gray), colorbar;
title('After morphological processing');
```

DEMO A: Instead of manual thresholding, perform automatic thresholding, similar to what you did in the previous lab.
Also improve the morphological processing. The goal is to get an image of the cell kernels with smooth borders and without holes, noise and small objects, see Figure 3. You will probably need the MATLAB function `bwareaopen`.

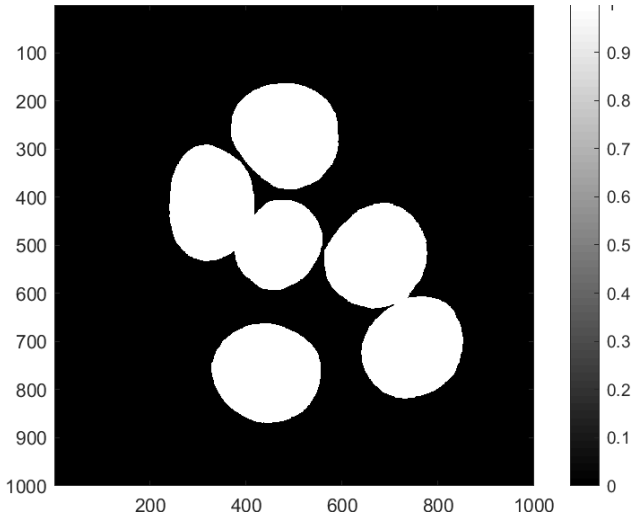

Figure 3: Desired binary image.

3.2 Watershed segmentation

The third part of the MATLAB script `SegmentCellsPrel.m` is shown below. The distance transform is calculated within the cells. Then some processing gives a suitable "landscape".

```
% Compute the distance transform within the cells
% ================================================
bw = im1bTmorph;
D = bwdist(~bw);
figure(8), imagesc(D); axis image, colormap(jet), colorbar;
title('Distance transform of ~bw');
colormap(jet), colorbar;

% Multiply the distance transform with -1
% and set pixels outside to the min-value.
% This is the desired landscape.
% ========================================
Dinv = -D;
Dinv(~bw) = min(min(Dinv));
figure(9), imagesc(Dinv), axis image, colormap(jet), colorbar;
title('Landscape 1');
colormap(jet), colorbar;
```

Execute `SegmentCellsPrel.m` in MATLAB and verify the result in figure(8)-figure(9)!

DEMO B: Segment the cell kernels using the watershed algorithm. You also need to do some preprocessing with `imhmin` to prevent oversegmentation. The goal is to first get a landscape similar to Figure 4. and then an image with labelled individual cell kernels as in Figure 5.

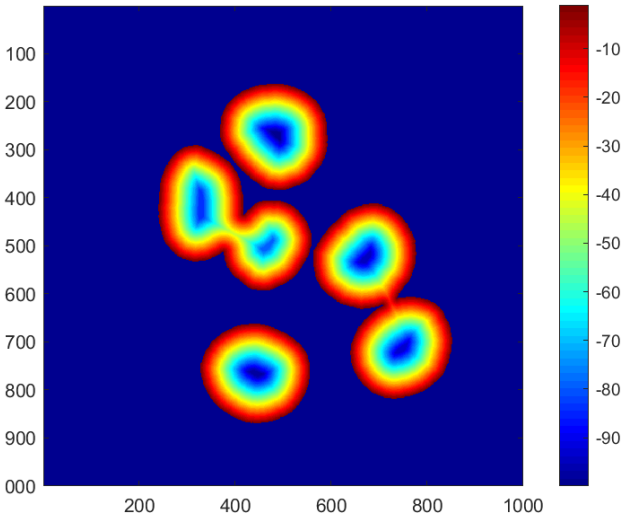

Figure 4: Distance map within cells treated with imhmin.

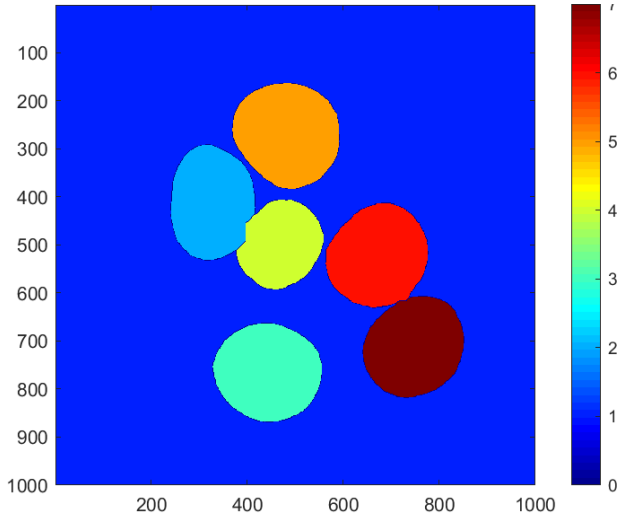

Figure 5: Desired labelling of the cell kernels.

The cytoplasm is located around the cell kernels. As mentioned previously, the hypothesis is that a pixel rather close to, but outside, the cell kernels contains cytoplasm and belongs to the closest cell kernel. Consequently, the cytoplasm can be determined by using the distance transform.

DEMO C: The desired cytoplasm "landscape" can be produced by first making a distance map outside the cell kernels and then limit the distance map to a suitable value. Use 100! The goal is to get an image as in Figure 6.

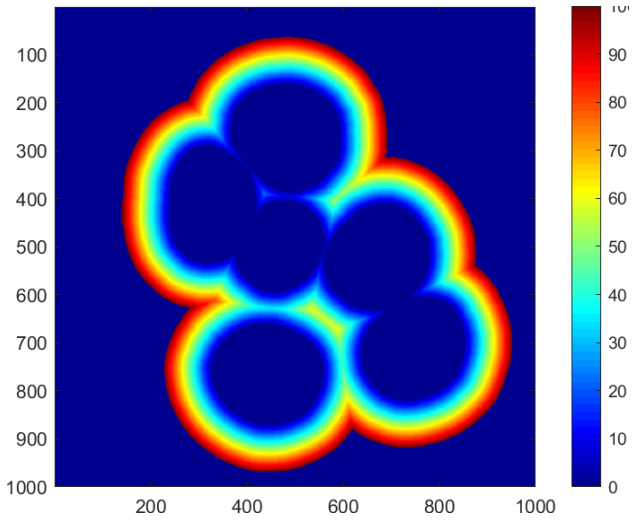

Figure 6: Desired landscape for the cytoplasms.

DEMO D: A combined landscape for the cells and cytoplasms can be produced by combining the cell distance map in Figure 4, the cytoplasm distance map in Figure 6 and the binary image in Figure 3. The goal is to get an image as in Figure 7.

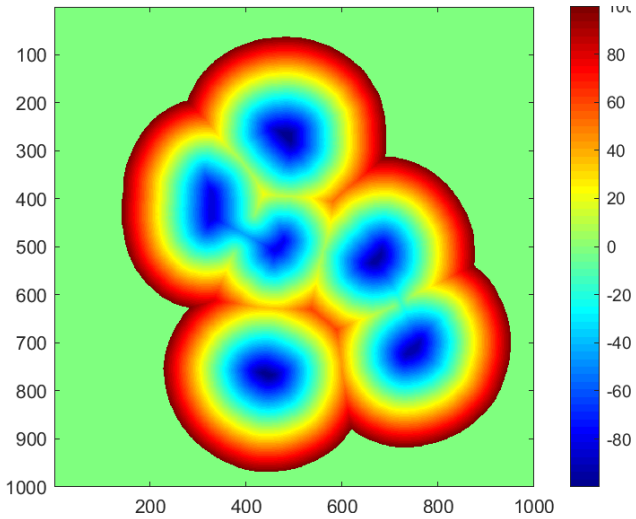

Figure 7: Combined landscape for the cells and cytoplasm.

Apply the watershed algorithm on your combined landscape for the cells and cytoplasms. The watersheds (dams) outline the cell cytoplasms. They can be extracted. They corresponds to the yellow borders in Figure 2.

Read and execute the MATLAB script `Overlay.m`. There, it is shown how to overlay a binary mask on the input color image. The example mask in `Overlay.m` is a magenta colored "M".

DEMO E: Overlay the outlines of the cell cytoplasms in *yellow* on the input color image, similarly as in Figure 2.

3.3 Localization of padlock-signals

As mentioned in the introduction, the laboratory assignment is inspired by an article. It tells about a 5×5 convolution filter which can be used to enhance local maxima. Such a filter might be the negative Laplacian filter,

$$-\Delta = -\frac{\partial^2}{\partial x^2} - \frac{\partial^2}{\partial y^2} = -\frac{\partial}{\partial x} * \frac{\partial}{\partial x} - \frac{\partial}{\partial y} * \frac{\partial}{\partial y}.$$

QUESTION 7: Construct such a filter based on the Sobel-operators. Two-dimensional convolution can be perfomed with the the `conv2`-command.
Sketch your negative Laplacian filter below!

-2.	-4	-4.	-4.	-2
-4	0.	8.	0.	-4
-4.	8.	24.	8.	-4
-4.	0.	8.	0.	-4
-2.	-4.	-4.	-4.	-2

/64

Your negative Laplacian filter can also be thought of as a matching filter for small circular objects, blobs. The filter contains a central circular positive area that matches blobs. The surrounding negative values make the filter DC-level independent. (More about this during the next laboratory assignment.)

DEMO F: The negative Laplacian filter acts as a padlock cell detector. The resulting image contains both positive and negative values. Smooth humps (or hills) are located at the padlock signals. Show this image!

DEMO G: Study the method to detect the padlock signals in Figure 8. Implement this method to detect the red padlock signals.

QUESTION 8: How many red padlock signals did you find?

49

DEMO H: Finish your program so that the red padlock signals in a selected cell is marked with circles. Let the color of the circles be *cyan*. The resulting image should look approximately as in Figure 2. The program should also print out the number of red padlock signals belonging to the selected cell.

QUESTION 9: How many red padlock signals did you find in the selected cell?

cell 6: 11 padlock signals

Questions:
Automatic thresholding for the red padlock signals? We just chose 40 by trial-and-error
Best way to generate the cyan circles?

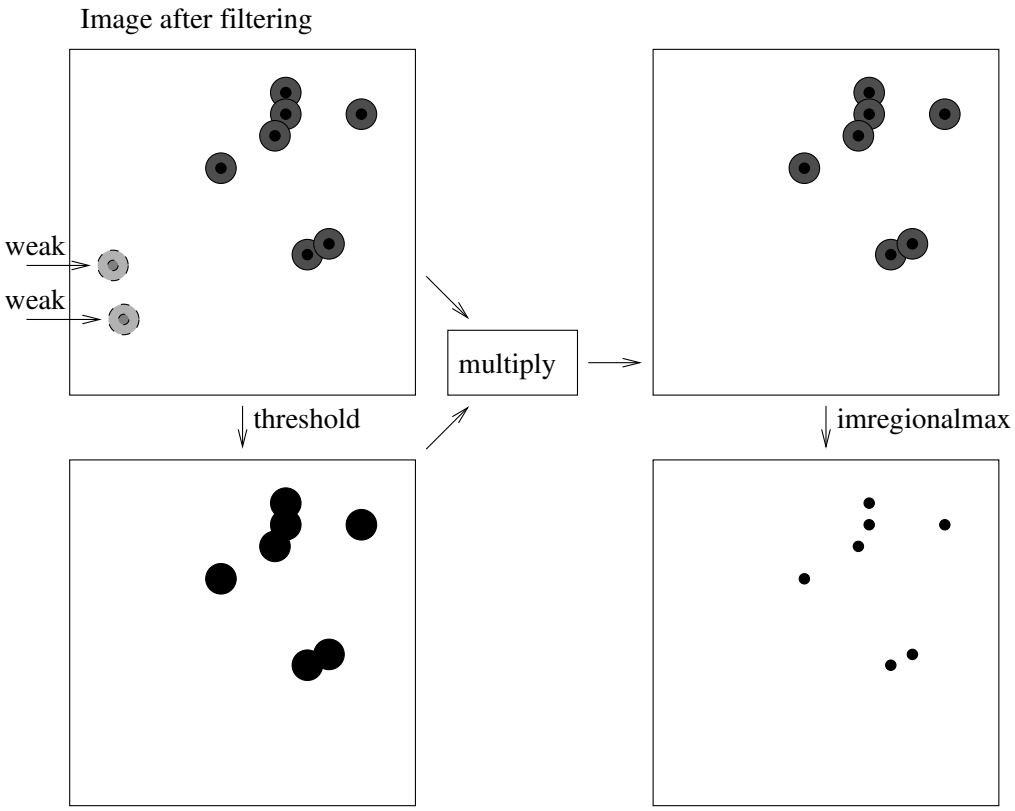

Figure 8: Padlock signal detection.