Practice 5. Morphological operators.

1. Practice Development

1.1. Binary image

To test all the exercises in this practice you can use the image blood5.tif (or a similar one).
First, show the image histogram and try to binarize it, choosing the best threshold. Is this value quite different than the value that use graythresh function?. Why?. Store this image for further uses.

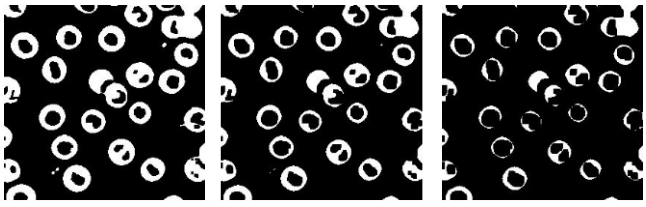
Original image Binarized image Negative image

1.2. Negative.m

Find the negative of the above binarized image and stored it.

1.3. IMERODE

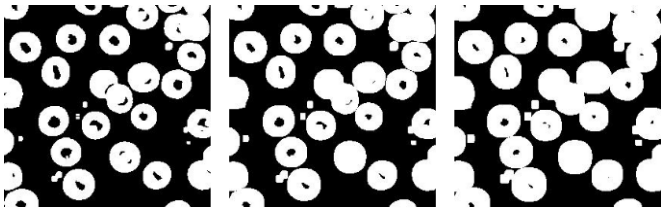
Test IMERODE Matlab function, with different parameters. This function needs a structuring element which will perform the erosion. Use STREL function.



Erosions with different kernel sizes 3, 5 and 7.

1.4. IMDILATE

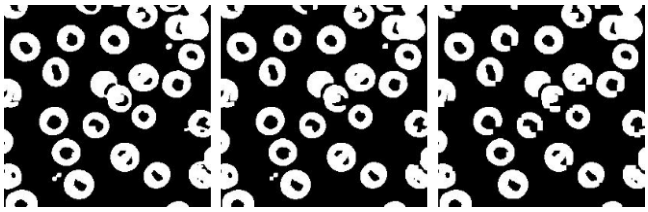
Test IMDILATE Matlab function, with different parameters. This function needs a structuring element which will perform the dilation. Use STREL function.



Dilations with different kernel sizes 3, 5 and 7.

1.5. IMOPEN

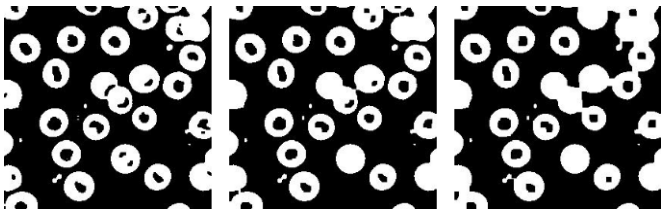
The opening function is the concatenation of an erosion and a dilation although Matlab performs it directly with IMOPEN. Try this function with different kernels.



Openings with different kernel sizes 3, 5 and 7.

1.6. IMCLOSE

The closing function is the concatenation of a dilation and an erosion although Matlab performs it directly with IMCLOSE. Try this function with different kernels.



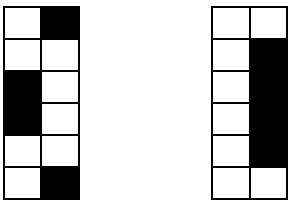
Closings with different kernel sizes 3, 5 and 7.

Exercise 5-1. Characters detection with desired features.

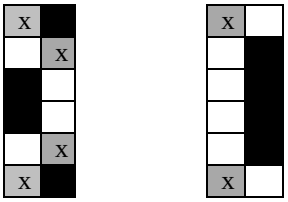
Supose that we want to detect those characters that contain vertical lines in text.tif image. To perform this operation it is necessary to erode the image using a vertical and narrow element proportional to the characters length (you can start with a 10x1 white pixel column as eroding element).
As a previous training, compare the results of the following functions on a binarized sample image (f):

a) fe = imerode(f, ones(10,1));
b) fo = imopen(f, ones(10,1));
c) fd = imdilate(fe, ones(10,1));
d) fobr = imreconstruct(fe,f);

So, the exercise consists in:
1. Binarization of the input image.
2. Find those characters that contain a pattern like one of the following two samples:
Find both cases:
a) only characters that match with the white pixels of the pattern
b) those characters that match with the white AND the black pixels of the pattern



3. Find those characters that match with white AND black pixels of the next patterns, but taking into account don't care pixels X (marked in gray with a X).



4. Present the results as an output image after each morphological operation you apply
5. Show some samples of characters that satisfices the conditions.

IMPORTANT: The use of *bwhitmiss* function is not allowed, except to compare with your results.

Exercise 5-2. Counting objects in a binary image.

Using blood5.tif, implement a program in Matlab to answer the following questions:

1. Count the number of cells in the image.
2. Find the center of mass of each cell and mark them with '*'
3. Find the rectangularity coefficient R of each cell, using $R = A_0/A_R$ where A_0 is the object area and A_R is the area of the minimum enclosing rectangle. This coefficient will get its maximum value (1.0) for rectangular objects.
4. Find the circularity coefficient C for each cell, using $C = P^2/A$, where P is the perimeter of object with area A. This coefficient will get its minimum value (4*pi) for round shapes.
5. Apply your solution to the image bloodcells.tif without changes. Are the results as expected?
6. Modify your solution to get good results in both images. Test with other images showing cells or grains (like "*arros.tif*").

To do this exercise you can use these indications:

a) binarize the image,
b) through mathematical morphology operators try to separate as much as possible the cells,
c) fill every cell with imfill function,
```
>> g = imfill(f, 'holes');
```
d) If you consider it necessary, fill every cell with imfill function,
```
>> g = imfill(f, 'holes');
```
e) If you consider it necessary, you can remove those elements that are too much closer to the image borders:
```
>> g = imclearborder(f, 8);
```
f) use the call
```
[L, n] = bwlabel (f)
```
This function returns the set of objects (L) that are in an image as a labelled image with the number of regions (n).
g) You can explore the scope of regionprops function, but it is not mandatory its use to solve the practice.
h) to write text over an image you can use:
```
imshow(f)     % the image to superimpose text
```

```
hold on
plot(x,y,'Marker','*','MarkerEdgeColor',i,'MarkerFaceColor',j,'MarkerSize',z);
```

where 'i', 'j' and 'z' are the values of plot modifiers.

Extra bonus 1. Pay attention on binarization threshold, because it is in the root of most of problems. Try to figure out a method to calculate the most suitable threshold. Also, try to improve your solution by smoothing the image in gray levels before binarizing.

Extra bonus 2. Test the watershed MATLAB function with the cells sample images.