Signal and image processing: Assignment 7

Mariuta Nicolae 23-10-2014

Exercise 1.

- Each of the morphological transformations dilation and erosion requires an image and a second matrix, the structuring element with different sizes, which is used much like the filters, applied over the entire image.
- A big difference from the filtering is that structuring element consists of binary pixels, with values of 1 and 0 and applied differently. The pixels of 1 build the shape of the structuring element. It can take different shapes, like cross, diamond, square, etc.
- For erosion, structuring element is only applied for positive pixels(foreground) and if any of the values of 1 in the structuring element is applied over 0 pixel, then the center of the mask becomes 0
- For dilation, structuring elements is only applied for negative pixels(background) and if the area where the structuring element is applied over 0 pixel, then the center of the mask becomes 1
- Each morphology has different effects over the image (usually binary), the erosion removes the edges of objects and links between them while the dilation makes the objects become larger and thickens the connection between items, sometimes creating a single object.
- Used together, with different shapes and sizes of structuring elements, several other morphological transformations can be obtained: opening, closing, TopHat, BottomHat, etc.

Exercise 2.

```
%2 opening
I = load('TestImageQn2.mat');
I = I.TestImageQn2;
cross = strel('diamond',1);

I1  = imerode(I,cross);
I2  = imdilate(I1,cross);

subplot(1,3,1);imshow(I);title('image');
subplot(1,3,2);imshow(I1);title('erode');
subplot(1,3,3);imshow(I2);title('dilate');
```

Fig. 1 Code for 'opening' the test image

I have loaded the image in Matlab, created the structuring element with diamond shape using *strel* function and then used function *imerode* and *imdilate* to make the 'opening' of the image, following the definition in Table 8.1 from the course book.

So, for the opening it is first applied the erosion and then dilatation by using the same structuring element. According to Matlab documentation on the website, the second parameter of function *strel* is defining the radius of the diamond, so for radius 1, the structuring element is a 3x3 diamon, so a cross with 5 pixels having value 1.

Fig. 2 Results for applying the opening to the test image, first apply erosion and then dilate

In the first step, most the white lines which are outside the blob disappear, and after applying the dilation, the blob is thickened and connected to the points that are outside. After erosion there is a pixel in the right that is left alone but that happens because, if looking in the original image there is actually a cross into which the structuring element can fit, and after dilation the point turns again into a cross.

Fig. 3 Results for making the opening using size 2 and size 3 structuring element

For blob detection, size 2 for the diamond (actually 5x5) is the best for removing all the branches that are outside the blob, which will probably only keep 1 or 2 white points in middle of the blob and then expand it using the dilation. Size number 3 for diamond (7x7) is too large and there is no such large square in the image, so it will result a black image.

```
%%
%2 closing
I = load('TestImageQn2.mat');
I = I.TestImageQn2;
cross = strel('diamond',1);

I1  = imdilate(I,cross);
I2  = imerode(I1,cross);

subplot(1,3,1);imshow(I);
subplot(1,3,2);imshow(I1);
subplot(1,3,3);imshow(I2);
```

Fig. 4 Code for making the closing operation

For the closing operation, I did almost the same as for the ‘opening’, just I have reversed the order that dilation and erosion are applied to apply the definition in table 8.1.

Fig. 5 results for making the closing over the test image

By running the code, after the dilation all the white areas are thickened and most of the white areas are connected. After erosion, the object is thinned again but still thicker than in the original image.

Fig. 6 Results for making the closing using size 2 and size 3 structuring element

By increasing the side of the diamond structuring element, the details of the object disappear until it becomes a single blob.

As for conclusion, the opening is removing some details of the object and turns it into a small blob where is most important area is located while the closing is connecting the object to details that are outside the main blob.

I think that opening can be useful to remove some noise in the image, and only keep the main points of interest while the closing can be useful to recover the shape of objects if the image is getting very blurred.

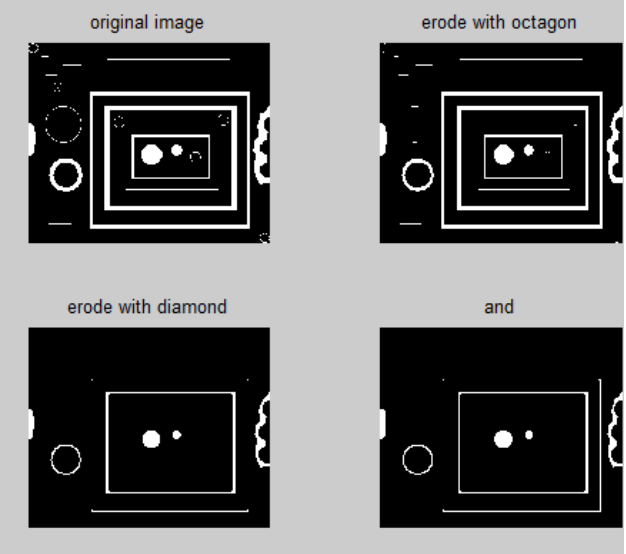
Exercise 2.

```
%3 hit and miss
I = imread('blobs.png');
B1 = strel('octagon',3);
B2 = strel('diamond',2);

I1  = imerode(I,B1);
I2  = imerode(I,B2);
Im  = and(I1,I2);

subplot(2,2,1);imshow(I);
subplot(2,2,2);imshow(I1);
subplot(2,2,3);imshow(I2);
subplot(2,2,4);imshow(Im);

%%
% 3 top hat
I = imread('blobs.png');
B1 = strel('octagon',3);

I1  = imerode(I,B1);
I2  = imdilate(I1,B1);
Im  = I - I2;

subplot(2,2,1);imshow(I);
subplot(2,2,2);imshow(I1);
subplot(2,2,3);imshow(I2);
subplot(2,2,4);imshow(Im);

%%
%3 buttom hat
I = imread('blobs.png');
B1 = strel('octagon',3);

I1  = imdilate(I,B1);
I2  = imerode(I1,B1);
Im  = I2 - I;

subplot(2,2,1);imshow(I);
subplot(2,2,2);imshow(I1);
subplot(2,2,3);imshow(I2);
subplot(2,2,4);imshow(Im);
```

Fig. 7 Code for applying Hit-and-miss, TopHat and BottomHat

For implementing these morphological transformations, I have chosen not to use Matlab functions to make them directly, but to use function *imdilate*, *imerode* and mathematical operations necessary to obtain the definitions in Table 8.1 from course book. By doing like this, I had better understanding of how these transformations work.

After these, I displayed images in all the stages of transformations to see how they work.



For doing 'hit and miss' with structuring elements *diamond* and *line* I removed thinner lines and kept the thick ones. Hit and miss applies 2 different structuring elements and keep only the common white areas remained after both erosions. It can be useful to remove some types of objects and shapes which have to be removed from the picture. I think it can actually be created more than two structuring elements and after that make *and* between them to keep only objects that are useful for the application.

Fig. 8 Hit and miss

Fig. 9 Top hat and Bottom hat transformations

For the image blobs I did not find some important features by applying Top hat or Bottom Hat so I applied them for *lena.tiff* image and obtained different features for them. The Top hat transformation returns elements that are smaller than the structuring element (diamond size 3) and areas that are brighter than the surroundings (because of subtracting the opening) while the bottom hat transformation returns the items that are smaller than the structuring element and areas that are darker than the surroundings(because of subtracting the closing).

Exercise 4.

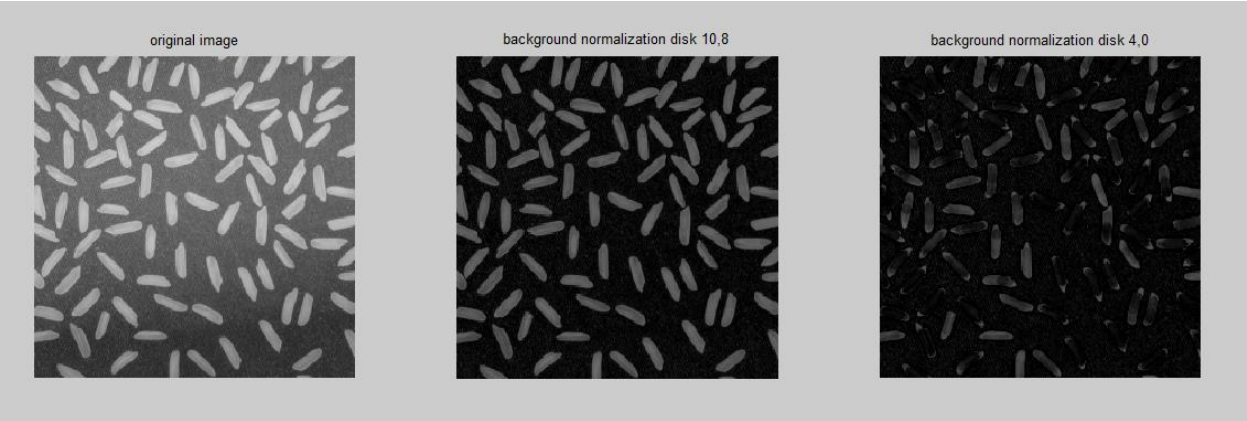


Fig. 10 Background normalization using different sizes of disk

I have done the background normalization using the same code as in Exercise 3 for Top Hat but changed the structuring element with *disk* shape that has 2 parameters: first for radius and the second one for determining the shape of disk.

I have obtained the best results for values (*10,8*) which created the background around the cells to have an uniform color, which is darker but helps to see the cells shape easier. For lower values of parameters, the cells are hard to distinguish because areas of the cells got the background color.

I achieved this because as I have found at Exercise 3, the Top Hat transformations keeps areas that are brighter than the surroundings. For small disk size, the shape of cells is no longer preserved because the cell areas are no longer larger than the structuring element. For even lower values of disk size, only the very blurred edges of cells are still visible.

Exercise 5.

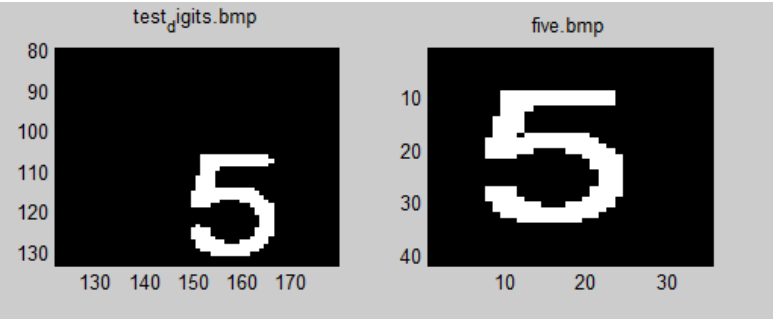


Fig. 11 Digit with number 5 from test_digits image which is zoomed and digit 5 from five.bmp

By simply using ‘hit and miss’ morphology having the structuring element the picture ‘*five.bmp*’ it will not work on detecting digit 5 in the *test_digits* image because it does not have exactly same shape(same white pixels) as the structuring element. The size of structuring element is same as in the *test_digits* but because it does not have exactly same pixels, it will not find the digit because *test_digits* does not have digit 5 to fit into structuring element

```
%5 hit-miss   digit 5
I = imread('test_digits.bmp');
B1 = imread('five.bmp');

%dilation of B1
b = strel('disk',1,0);
B1d = imdilate(B1,b);

B2 = ~B1d;
Im  = bwhitmiss(I,B1,B2);

[i,j] = find(Im == 1);
```

Fig. 12 Code for detecting digit 5

For solving the issue, I had to apply dilation over the structuring element to make it thicker so the digit in *test_digits* will fit into it. After that I took the complementary of the structuring element and used function *bwhitmiss* to apply the hit and miss morphology.

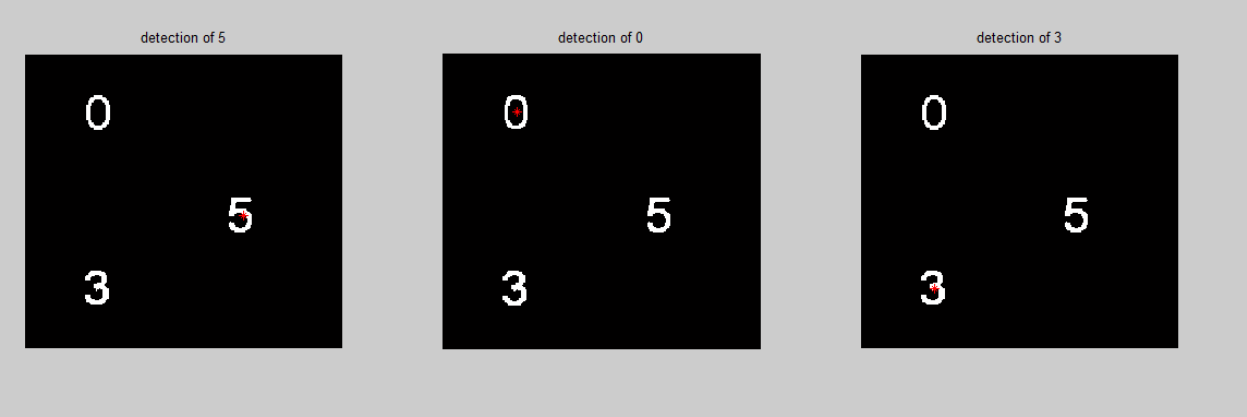


Fig. 13 Detection of digits in test_digits

By doing the dilation before applying the structuring element, I have managed to detect all the digits in the image.

This makes me think that hit and miss can have great applications in detecting type characters in a photo that is taken or maybe even handwriting by making samples of structuring elements for each character.

But unfortunately, after rotating and scaling the test_digits image, the code that I wrote could no longer detect the digits, even if I scaled and rotated the structuring elements with same values.

This means that lot more features have to be obtained in order to detect all the characters in a photo taken to the package of a product (for translating or searching information on web). The characters can be written using different styles, formats so many checks have to be made. A database with structuring elements containing characters of all shapes, sizes and angles would be too large to make this morphology to be useful. For handwriting detecting it would be even harder to detect characters because each person has different shapes of characters, so even more calculations have to be made, maybe detecting key points and machine learning algorithms.

Exercise 6.

```
I = imread('flowers.jpg');
subplot(2,2,1);imshow(I);title('original image');
%convert to greyscale
I = rgb2gray(I);
subplot(2,2,2);imshow(I);title('greyscale image');
%blur the image
filter = fspecial('gaussian',[250 250],150);
I =I - imfilter(I,filter);
subplot(2,2,3);imshow(I);title('blurred image');
%edge detection
se = strel('disk',3,0);
Amax = imdilate(I,se);
Amin = imerode(I,se);
Mgrad = Amax - Amin;
subplot(2,2,4);imshow(I);title('edge detection');
```

Fig. 14 Read image, convert to greyscale, blurr and edge detection

For the first stages I read the image and converted to greyscale using *rgb2gray* function. For blurring the image I have applied a Gaussian filter with size and sigma parameter large enough so the surroundings of the flower almost disappear. I did this mostly because in the final result I kept getting those drops of rain in background. By blurring the image enough, they disappeared.

In the next stage I applied edge detection morphology to obtain the edges of the flower which are easier to distinguish than the edges of the rain drops. For edge detection I used a disk shape structuring element with size 3. I have found this shape to be the best, with larger shape of structuring element, the flower petals disappear. Unfortunately, after making the edge detection details in the middle of the flower disappeared.

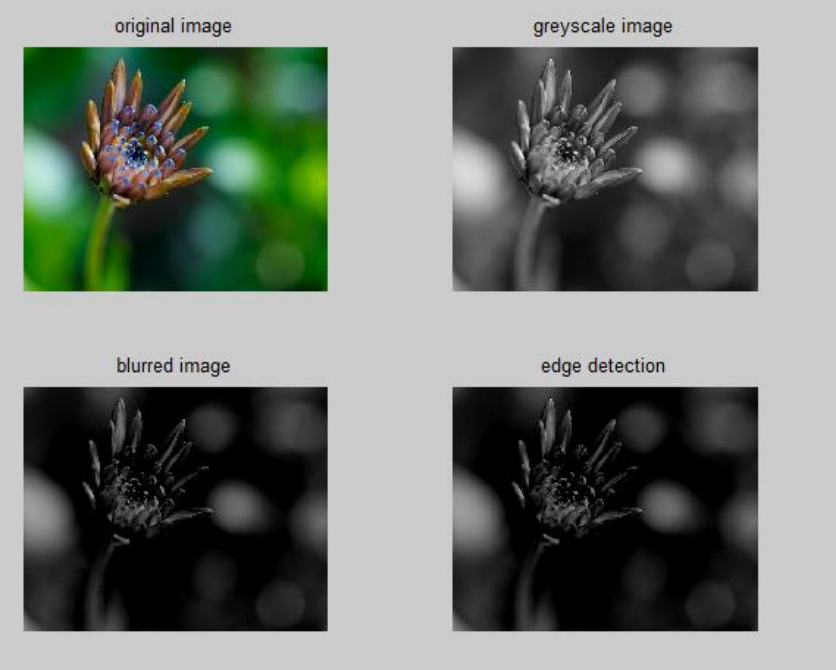


Fig. 15 Result for edge detection

```
%dilation and erodion
se = strel('disk',3,8);
Mgrad2 = imerode(Mgrad,se);
Mgrad3 = imdilate(Mgrad2,se);
subplot(1,2,1);imshow(Mgrad3);
%make image binary by using threshold
mask = im2bw(Mgrad3,0.1);

%apply mask
figure;
h = imshow(imread('flowers.jpg'));
set(h, 'alphadata',mask);
```

Fig. 16 Code for creating and applying the mask

For the last part of the code I tried to remove parts of the picture that are not important and only keep the focused flower so I can make the mask as good as possible. I tried to remove the edges of the picture and other details by applying an opening.

I have found that threshold limit of 0.1 to be the best, for lower value more details surrounding the flower appear and for larger values, the flower can no longer be distinguished.

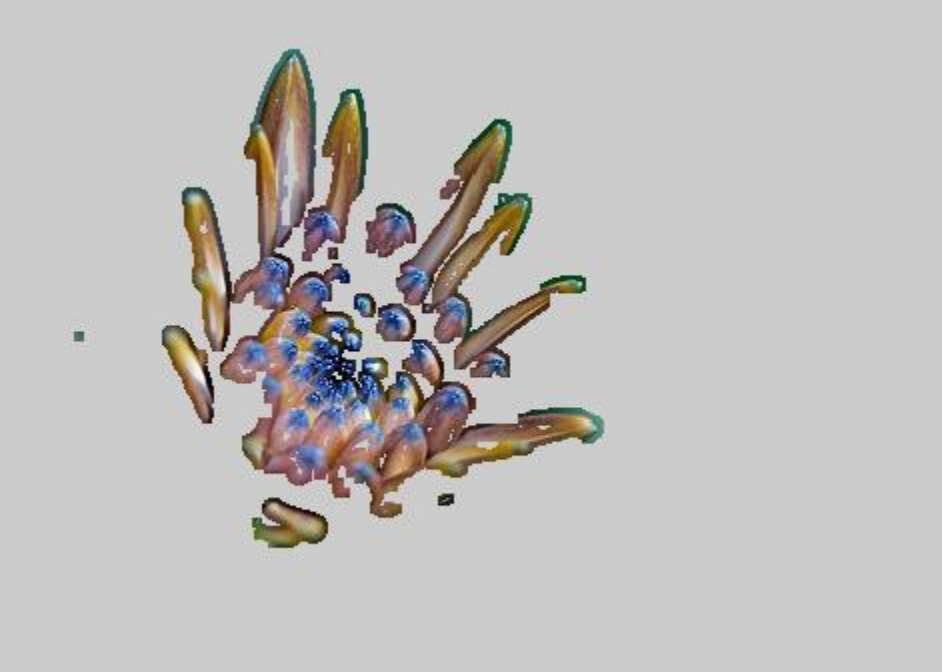


Fig. 17 Final result after applying the mask

Unfortunately, for in the final result many details of the flower disappeared but its shape can be recognized. I have tried to do some filling and thickening to obtain the whole image of the flower but I did not manage to do it. However, I think that the areas inside the petals of the flower have to be filled with higher pixel values so they do not longer disappear.