Labs 3 - Morphogical mathematics

Submitted by **Meldrick REIMMER, Selma BOUDISSA**

Contents

1 Problem 1 **3**
1.1 Load an image . . . 3
1.2 Binarize an image . . . 3
1.3 Effect of the erosion operator . . . 4
1.4 Others approaches . . . 5
1.4.1 Dilatation . . . 5
1.4.2 Closing . . . 6
1.4.3 Opening . . . 6
1.5 Method to measure the granulometry . . . 7

2 Problem 2 **7**
2.1 Erosion Gray . . . 7
2.2 Dilatation Gray . . . 8
2.3 Opening Gray . . . 8
2.4 Closing Gray . . . 9

3 APPENDIX **10**
3.1 Code for problem 1 . . . 10
3.2 Code for problem 2 . . . 11

1 Problem 1

Objective:

1.1 Load an image

Here the code to load the

```
%% Problem 1
%load an image
image=imread('./images/hubble.png');
```

load.m

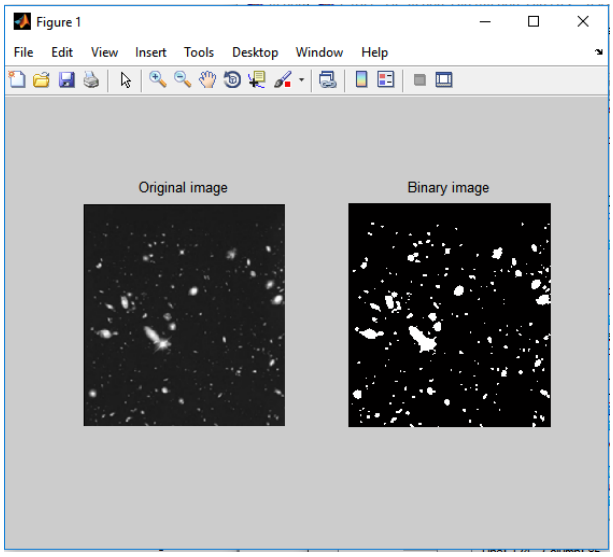
1.2 Binarize an image

Here after the code to binarize the original image.

```
%binarize an image
BW = im2bw(image, 0.2);
```

binarize.m

Comments: In the matlab code we use the im2bw which will convert the image to binary image, based on treshold. BW = im2bw(image, 0,2); The value 0,2 correspond to the level. The output image BW replaces all pixel in the input image with luminance greather than level with the value 1 (white) and replaces all other pixels with the value 0 (black). We chose a level of 0,2 and we obtain a better result. The figure below shows the result obtain by the binarization of the orginal image.



1.3 Effect of the erosion operator

We create a function erosion to reply to the question, here after the code. We had help from our classmate Antoine Merlet for an approach to this function.

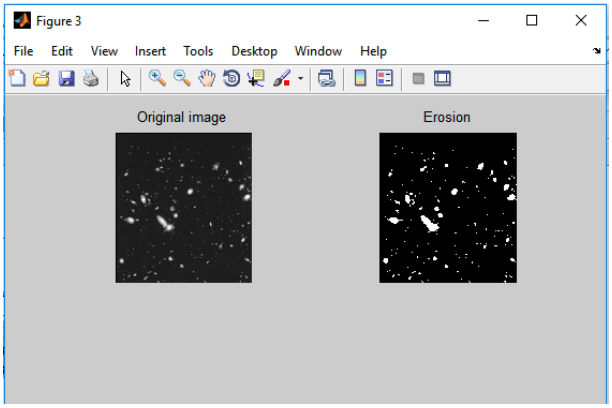
```
function retimage = erosion(image,pattern)
    % Retriveing respectivly image & pattern height and width
    [marginV,marginH] = size(pattern);
    [sizeV,sizeH]=size(image);
    % Calculating the number of pixel(s) taken by the Structural Element on each
    direction on
    % the filter
    marginV = (marginV - 1 )/2;
    marginH = (marginH - 1 )/2;

    % Creating the output
    retimage = zeros(size(image));

    % Parsing the image. To remove any out-of-bound problem,
    % we do not consider the outermost lines
    for i=1+marginV:sizeV-marginV
        for j =1+marginH:sizeH-marginH
            mycurrentmatrix = image(i-marginV:i+marginV,j-marginH:j+marginH);
            res1 = mycurrentmatrix & pattern;
            if sum(res1(:)) == sum(pattern(:))
               retimage(i,j)=1;
            end
        end
    end
    retimage=im2bw(retimage,0.5);
end
```

erosion.m

In the figure below the result of the erosion operator on the original image.



Explanations: Erosion shrinks the boundary of a binary image to be smaller. That is why we use the binary image for the erosion operator: resultEro = erosion (BW,pattern); Concerning the size of structuring element we use a cross pattern named pattern3C in the matlab code.

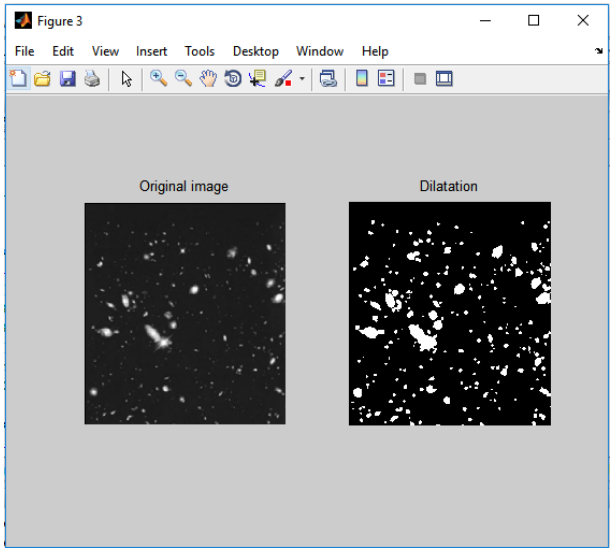
1.4 Others approaches

1.4.1 Dilatation

```
function retimage = dilatation(image,pattern)
    % Retriveing respectivly image & pattern height and width
    [marginV,marginH] = size(pattern);
    [sizeV,sizeH]=size(image);
    % Calculating the number of pixel(s) taken by the Structural Element on each
    direction on
    % the filter
    marginV = (marginV - 1 )/2;
    marginH = (marginH - 1 )/2;
    % Creating the output
    retimage = zeros(size(image));

    % Parsing the image. To remove any out-of-bound problem,
    % we do not consider the outermost lines
    for i=1+marginV:sizeV-marginV
        for j =1+marginH:sizeH-marginH
            mycurrentmatrix = image(i-marginV:i+marginV,j-marginH:j+marginH);
            res2 = mycurrentmatrix & pattern;
            if sum(res2(:)) > 0
               retimage(i,j)=1;
            end
        end
    end
    retimage=im2bw(retimage,0.5);
end
```

dilatation.m



Explanations: Dilatation expands the boundary to be larger. Usually the shrinkage or growth is by one pixel, but greater changes are possible with the recipe provided by the structuring element.

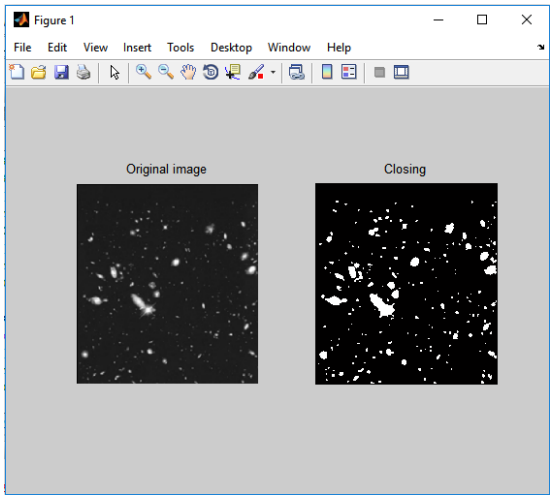
1.4.2 Closing

```
function retimage = closing(image,pattern)

    resultDila = dilatation(image,pattern);
    retimage = erosion(resultDila,pattern);

end
```

closing.m

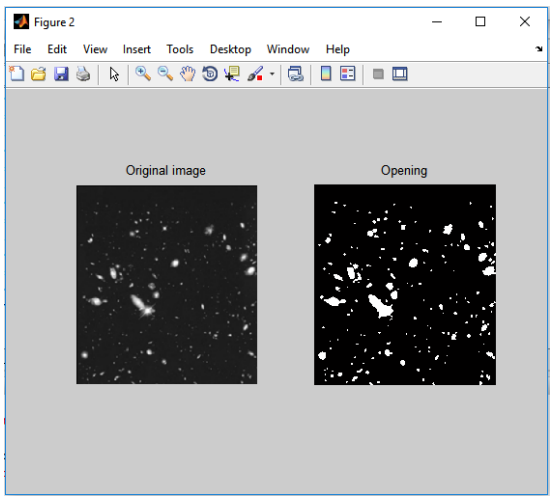


1.4.3 Opening

```
function retimage = opening(image,pattern)

    resultEro = erosion(image,pattern);
    retimage = dilatation(resultEro,pattern);
end
```

opening.m



1.5 Method to measure the granulometry

The granulometry method is a way to measure a particule size in an image. It determines the size distribution of object in an image without detecting each object first.

2 Problem 2

2.1 Erosion Gray

```
function retimage = erosionGray(image,pattern)
    % Retriveing respectivly image & pattern height and width
    [marginV0,marginH0] = size(pattern);
    [sizeV,sizeH]=size(image);
    % Calculating the number of pixel(s) taken by the Structural Element on each
    direction on
    % the filter
    marginV = (marginV0 - 1 )/2;
    marginH = (marginH0 - 1 )/2;
    special = 0;

    if sum(pattern(:)) ~= marginV0 * marginH0
        special = 1;
    end

    % Creating the output
    retimage = zeros(size(image),'uint8');

    % Parsing the image. To remove any out-of-bound problem,
    % we do not consider the outermost lines
    for i=1+marginV:sizeV-marginV
        for j =1+marginH:sizeH-marginH
            mycurrentmatrix = image(i-marginV:i+marginV,j-marginH:j+marginH);
            currentmin = Inf;

            if special
                for k = 1 : marginV0
                    for l = 1 : marginH0
                        if pattern(k,l)==1
                            if mycurrentmatrix(k,l) < currentmin
                               currentmin =  mycurrentmatrix(k,l);
                            end
                        end
                    end
                end
            else
                currentmin = min(min(mycurrentmatrix));
            end
            retimage(i,j) = currentmin;
        end
    end
end
```

erosionGray.m

2.2 Dilatation Gray

```
function retimage = dilatationGray(image,pattern)
    % Retriveing respectivly image & pattern height and width
    [marginVO,marginHO] = size(pattern);
    [sizeV,sizeH]=size(image);
    % Calculating the number of pixel(s) taken by the Structural Element on each
    direction on
    % the filter
    marginV = (marginVO - 1 )/2;
    marginH = (marginHO - 1 )/2;

    special = 0;

    if sum(pattern(:)) ~= marginVO * marginHO
        special = 1;
    end
    % Creating the output
    retimage = zeros(size(image),'uint8');

    % Parsing the image. To remove any out-of-bound problem,
    % we do not consider the outermost lines
    for i=1+marginV:sizeV-marginV
        for j =1+marginH:sizeH-marginH
            mycurrentmatrix = image(i-marginV:i+marginV,j-marginH:j+marginH);
            currentmax = 0;

            if special
                for k = 1 : marginVO
                    for l = 1 : marginHO
                        if pattern(k,l)==1
                            if mycurrentmatrix(k,l) > currentmax
                               currentmax =  mycurrentmatrix(k,l);
                            end
                        end
                    end
                end


            else
                currentmax = max(max(mycurrentmatrix));
            end
            retimage(i,j) = currentmax;
        end
    end
end
```

dilatationGray.m

2.3 Opening Gray

```
function retimage = openingGray(image,pattern)

    resultEro = erosionGray(image,pattern);
    retimage = dilatationGray(resultEro,pattern);

end
```

openingGray.m

2.4 Closing Gray

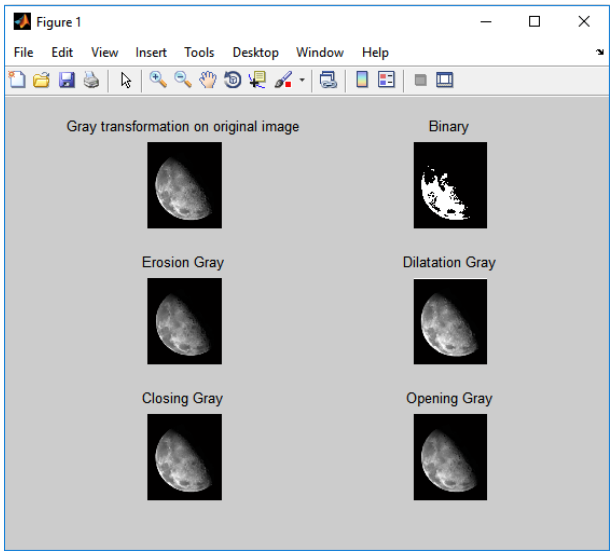
```
function retimage = closingGray(image,pattern)

    resultDila = dilatationGray(image,pattern);
    retimage = erosionGray(resultDila,pattern);

end
```

closingGray.m

In the figure below you will find the result of the different approach used. We took the same code as problem 1 and modified the original image to a transformation into gray level using the rgb2gray matlab function.



3 APPENDIX

3.1 Code for problem 1

```
%% LAB 3 - Morphological mathematics
% UE2-3 Image Processing
% Group: Meldrick Reimmer and Selma Boudissa

%% basic command
clc % clear command window
close all % close all the figures
clear all % clear all the variables

%% Problem 1
%load an image
image=imread('./images/hubble.png');

%binarize an image
BW = im2bw(image, 0.2);

% Structure choice
pattern3C = [0 1 0;1 1 1;0 1 0]; % code the pattern as matrix
pattern  = pattern3C;

%erosion operator
resultEro = erosion(BW,pattern);

%dilation operator
resultDila=dilatation(BW,pattern);

%opening operator
resultOpen = opening(BW,pattern);

%closing operator
resultClose = closing(BW,pattern);


% Displaying plot
figure()
subplot(3,2,1)
imshow(image);
title('Original image')
subplot(3,2,2)
imshow(BW);
title('Binary image')
subplot(3,2,3);
imshow(resultEro);
title('Erosion');
subplot(3,2,4);
imshow(resultDila);
title('Dilatation');
subplot(3,2,5);
imshow(resultClose);
title('Closing');
subplot(3,2,6);
imshow(resultOpen);
title('Opening');
```

prob1.m

3.2 Code for problem 2

```
%% LAB 3 - Morphological mathematics
% UE2-3 Image Processing
% Group: Meldrick Reimmer and Selma Boudissa

%% basic command
clc % clear command window
close all % close all the figures
clear all % clear all the variables

%% Problem 2
%load an image
image=imread('./images/moon.png');
% transform into gray level using rgb2gray
imageGray=rgb2gray(image);

%binarize an image
BW = im2bw(imageGray, 0.5); % we choose a level of 0.5 in the midway of black and
    white [0-1]

% Structure choice
pattern5M = ones(5);
pattern  = pattern5M;

%erosion operator
resultEro = erosionGray(imageGray,pattern);

%dilation operator
resultDila=dilatationGray(imageGray,pattern);

%opening operator
resultOpen = openingGray(imageGray,pattern);

%closing operator
resultClose = closingGray(imageGray,pattern);


% Displaying plot
figure()
subplot(3,2,1)
imshow(imageGray);
title('Gray transformation on original image')
subplot(3,2,2)
imshow(BW);
title('Binary')
subplot(3,2,3);
imshow(resultEro);
title('Erosion Gray');
subplot(3,2,4);
imshow(resultDila);
title('Dilatation Gray');
subplot(3,2,5);
imshow(resultClose);
title('Closing Gray');
subplot(3,2,6);
imshow(resultOpen);
title('Opening Gray');
```

prob2.m