Lab. Practice #4

Berk ARSLAN

Morphological Operators

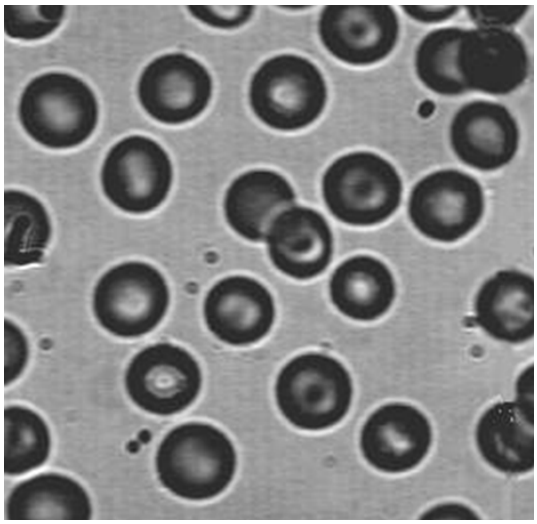
Exercise 4-1. Counting objects in a binary image

The function has basic two part which are to get image ready and calculation of cells.

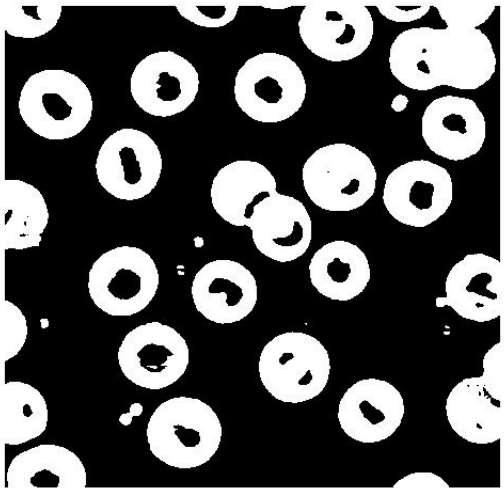
The first step including basic morphological operations. First of all, it gets complemantary of binarized image then fills the holes and uses erosion and dilation operations which are opening operation together. At the end of the part fills the holes again. With crop input we have two opportunities to choose crop image and get rid of cells on the edge or use process for whole cells.

The second part starts with getting perimeters of cells. At the beginning the image eroded and subtracted from the original image then the absolute of the solution gives the perimeters of cells. In the next step, each cell labeled and colorized according to label. Standard deviation of cells' masses calculated to eleminate out of focus cells with regionprops function.

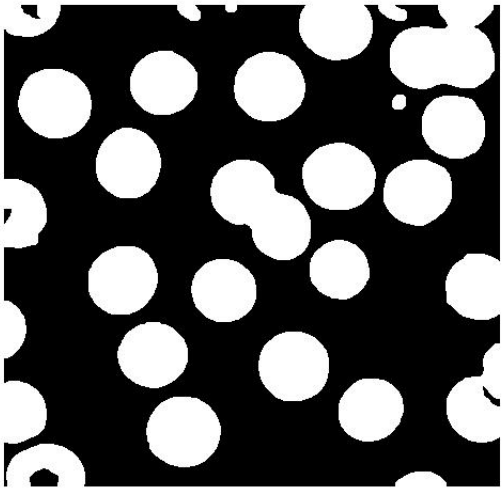
The loop is used to calculate the center of mass and coefficients. For each label, whole image is scanned and if it has label, then maximum and minimum values of boundary box, mass and perimeter of cells. Finally mass of each cell is checked and if it is bigger than the standard deviation, center of mass, circularity and rectengularity coefficients are calculated and boundary box is printed with center of mass points.



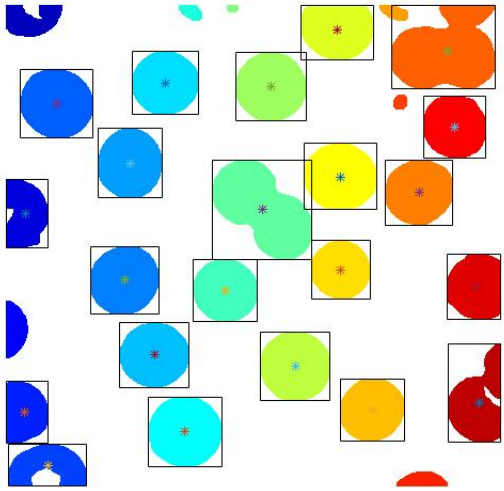
A



B



C



D

A- Original image, B- Opened image C- Filled Image D- Labeled and Calculated Image

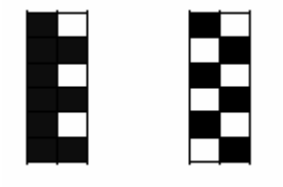
Exercise 4-2. Characters detection with desired features

The purpose of exercise is detecting the letters with given filter masks. Hit or Miss transform is the solution of problem and the following equation is the hit or miss operation;

$$A \odot B = (A \ominus C) \cap (A^c \ominus D)$$

The operation that detects a given configuration (or pattern) in a binary image, using the morphological erosion operator and a pair of disjoint structuring elements. That means the feature eroded with foreground and complementary of feature eroded with background gives the wanted patterns in image. Final step is reconstructing the image with hit or miss output and original image so, the letters which have the fature are appears.

The following features are described as willed;

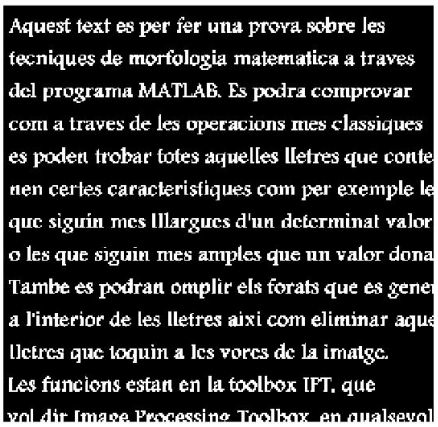


Firstly, to get the patterns, image is binarized (E) then gets complementary (F). The image is eroded with filter (G) for hit and the complementary of both eroded (H) for miss. On the figure (J), features are detected with hit-or-miss method and finally reconstructed in (K) figure with original image.

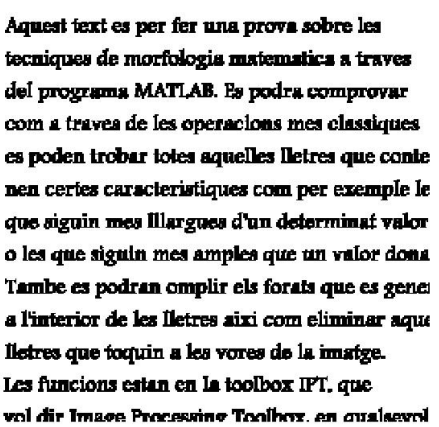
However when the patterns are tried in the code, features could not detected. To proof that it is working, another feature is tried and detected at output.

Tested feature named as filter3;

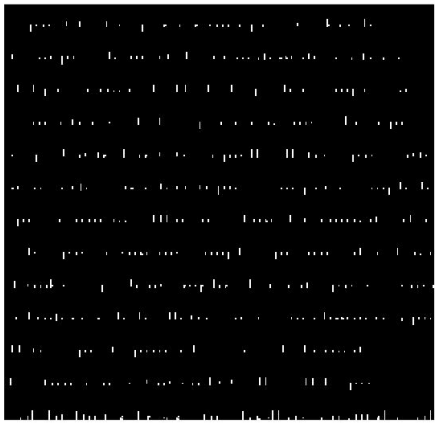
```
1 1
1 0
1 1
1 0
1 0
1 1
```


Aquest text es per fer una prova sobre les
tecniques de morfologia matematica a traves
del programa MATLAB. Es podra comprovar
com a traves de les operacions mes classiques
es poden trobar totes aquelles lletres que conte
nen certes caracteristiques com per exemple le
que siguin mes llargues d'un determinat valor
o les que siguin mes amples que un valor dona
Tambe es podran omplir els forats que es gene
a l'interior de les lletres aixi com eliminar aque
lletres que toquin a les vores de la imatge.
Les funcions estan en la toolbox IPT, que

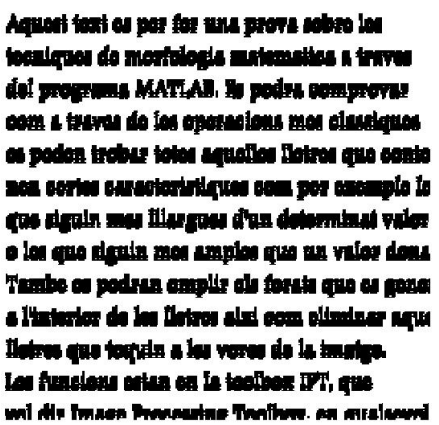
E


Aquest text es per fer una prova sobre les
tecniques de morfologia matematica a traves
del programa MATLAB. Es podra comprovar
com a traves de les operacions mes classiques
es poden trobar totes aquelles lletres que conte
nen certes caracteristiques com per exemple le
que siguin mes llargues d'un determinat valor
o les que siguin mes amples que un valor dona
Tambe es podran omplir els forats que es gene
a l'interior de les lletres aixi com eliminar aque
lletres que toquin a les vores de la imatge.
Les funcions estan en la toolbox IPT, que

F

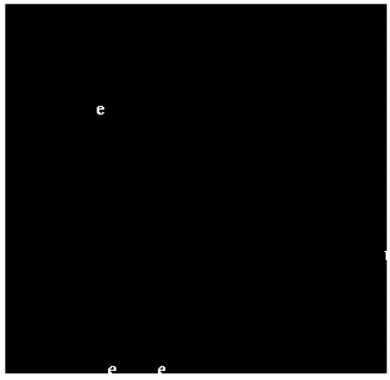


G


Aquest text es per fer una prova sobre les
tecniques de morfologia matematica a traves
del programa MATLAB. Es podra comprovar
com a traves de les operacions mes classiques
es poden trobar totes aquelles lletres que conte
nen certes caracteristiques com per exemple le
que siguin mes llargues d'un determinat valor
o les que siguin mes amples que un valor dona
Tambe es podran omplir els forats que es gene
a l'interior de les lletres aixi com eliminar aque
lletres que toquin a les vores de la imatge.
Les funcions estan en la toolbox IPT, que

H

J



K

Codes

Practice 4

```
function [ n,center,R,C ] = practice4( image , crop )
% n = number of cells
% center = center of mass point of each cell
% R = rectangularity coefficient
% C = circularity coefficient
figure, imshow(image)

% binarizing
image_threshold = graythresh(image)
image_bin = im2bw(image, image_threshold);
figure, imshow(image_bin)

% inverting the binarized image
image_comp_bin = ~image_bin;
figure, imshow(image_comp_bin)

% first filling operation
image_comp_fill = imfill(image_comp_bin, 'holes');
figure, imshow(image_comp_fill)

% imopen worked better as imclose
struct_element = strel('disk', 6);
image_fill_open = imopen(image_comp_fill, struct_element);
figure, imshow(image_fill_open)

% second filling operation
image_2fill = imfill(image_fill_open, 'holes');
figure, imshow(image_2fill)

if (crop)
    % clearing the border
    image_ready = imclearborder(image_2fill, 6);
```

```
    figure, imshow(image_ready)
else
    image_ready = image_2fill;
end

% erode one more time and subtract from previous and we get perimeter
% and label again
se = strel('square', 3);
image_ready_erode = imerode(image_ready, se);
image_perimeter = logical(abs(imsubtract(image_ready_erode, image_ready)));
[perimeter_label,n_perimeter] = bwlabel(image_perimeter);

% % just a test to find and color the disks
% the number of cells in the picture
[L,n] = bwlabel(image_ready);
RGB = label2rgb(L);
figure, imshow(RGB)
hold

%get sizes of label image
[sizex, sizey] = size(L);
%define as zeros the outputs
R = zeros (1,n);
P = zeros (1,n);
C = zeros (1,n);
mass = zeros (1,n);
center = zeros(2,n);

%set standard deviation of area
deviation = regionprops(image_ready, 'Area');
st_dev = std([deviation.Area]);

%count for each label
for label = 1:n
    %determine the variables
    x = 0;
    y = 0;
    minX = 0;
    maxY = 0;


    firstX = 1;
    firstY = 1;
    %for each image cell
    for i = 1:sizex
        for j = 1:sizey

           % if the pixel is labeled determine the boundary box's minimum values
            if (L(i,j) == label)
                 if (firstX && minX < i)
                     minX = i;
                     firstX = 0;
                 end
                 if (firstY || minY > j)
                     minY = j;
                     firstY = 0;
                 end
                % sum of all pixels for each label
                mass(label) = mass(label) + 1;
                % count cordinates
                x = x + i;
                y = y + j;
                % determine maximum values of boundary box
                maxX = i;
```

```
                if (maxY < j)
                    maxY = j;
                end
            end
            % if the pixel is labeled add one to perimeter of label
            if(perimeter_label(i,j) == label)
                P(label) = P(label) + 1;
            end

        end
    end
    % if the cell is big enough
    if (mass(label) >= st_dev)
        % calculate center of mass
        center(:,label) = [ x/mass(label) y/mass(label) ];
        % rectengularity coefficient
        R(label) = mass(label) / ((maxY-minY) * (maxX-minX));
        % circularity coefficient
        C(label) = P(label)^2 / mass(label);
        % print boundary box and center of mass
        rectangle('Position',[minY,minX,maxY-minY,maxX-minX]);
        plot(center(2,label),center(1,label),'Marker','*')
    end
end
```

Text Find

```
function [ final ] = textfind( text , filter )

%threshold image
text_thr = graythresh(text);

%binarized input image
text_bw = im2bw(text, text_thr);

%print binarized text
figure
subplot(1,2,1)
imshow(text_bw);

%print complemntary of text
text_comp = ~ text;
subplot(1,2,2);
imshow(text_comp)

%complementary of filter
filter_comp = ~filter;

%print erosion the first pattern
figure
subplot(1,2,1)
hit = imerode(text_bw, filter);
imshow(hit);

%print erosion the second pattern from complementaries of text and filter
```

```
subplot(1,2,2)
miss = imerode(text_comp, filter_comp);
imshow(miss)

%print hit and miss output
figure
HitorMiss = hit & miss;
imshow(HitorMiss)

%print final image
figure
final = imreconstruct(HitorMiss, text_bw);
imshow(final)

end
```

Text Find Calling Script

```
clear all; clc; close all;
image = imread('text.tif');

filter1 = [0 1;
           0 0;
           0 1;
           0 0;
           0 1;
           0 0];


filter2 = [0 1;
           1 0;
           0 1;
           1 0;
           0 1;
           1 0];

filter3 = [1 1;
           1 0;
           1 1;
           1 0;
           1 0;
           1 1];


textfind(image, filter1);
textfind(image, filter2);
textfind(image, filter3);
```