EX5 - Simple Image Segmentation

## Thresholding with a fixed (global) threshold

clc

clear all

t\_s = '\fontsize{12}\color{black}\bf';

ft\_s = '\fontsize{18}\color{black}\bf';

s\_s = '\fontsize{12}\color{gray}\rm';

img = rgb2gray(imread("20191121\_072040.jpg"));

h = subplot(2,3,1);

imshow(img);

formattedText = {strcat(t\_s,'Image 1 (I1)');};

title(h, formattedText);

h = subplot(2,3,4);

imhist(img);

formattedText = {strcat(t\_s,'Hist(I1)');};

title(h, formattedText);

h = subplot(2,3,2);

binary\_img = img > 210;

binary\_img = 1 - binary\_img;

imshow(binary\_img);

formattedText = {strcat(t\_s,'Binarization based on histogram'); strcat(s\_s,'B1 t=210') };

title(h, formattedText);

h = subplot(2,3,5);

[labels, N\_object] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B1'); strcat(s\_s,'N =',string(N\_object))};

title(h, formattedText);

h = subplot(2,3,3);

[binary\_img\_otsu, threshold] = otsu\_method(img);

imshow(binary\_img\_otsu);

formattedText = {strcat(t\_s,'Otsu Binarization'); strcat(s\_s,'B1O t=',string(threshold)) };

title(h, formattedText);

h = subplot(2,3,6);

[labels, N\_object] =bwlabel(binary\_img\_otsu);

labels\_rgb=label2rgb(labels);

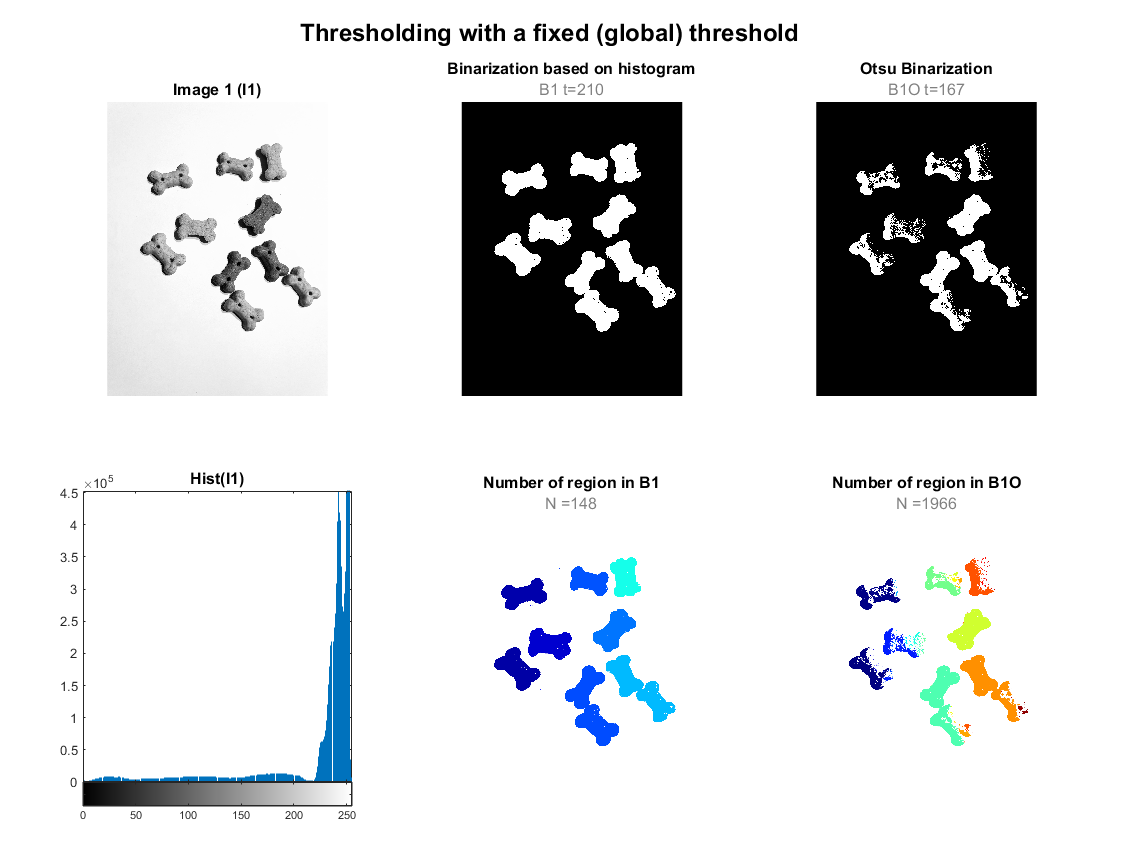
imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B1O'); strcat(s\_s,'N =',string(N\_object))};

title(h, formattedText);

formattedText = {strcat(ft\_s,'Thresholding with a fixed (global) threshold');};

sgtitle(formattedText);



Threshold

## 

Threshold

The binarization result are quite good, with interest object well extracted from the background. The presence noise (visible with small dots), contourns not well defined and holes in the object can be easily solved with further morphological operations.

## 3) Effect of preprocessing on binarization

**Gaussian Noise**

figure

% Original image

h = subplot(3,4,1);

imshow(img);

formattedText = {strcat(t\_s,'I');};

title(h, formattedText);

h = subplot(3,4,2);

imhist(img);

formattedText = {strcat(t\_s,'hist(I)');};

title(h, formattedText);

h = subplot(3,4,3);

binary\_img = img > 210;

binary\_img = 1 - binary\_img;

imshow(binary\_img);

formattedText = {strcat(t\_s,'B t=210');};

title(h, formattedText);

h = subplot(3,4,4);

[labels, ~] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B');};

title(h, formattedText);

% Noise

h = subplot(3,4,5);

img\_noise = imnoise(img,'gaussian',0.05);

imshow(img\_noise);

formattedText = {strcat(t\_s,'Gaussian noise image - I\_n');};

title(h, formattedText);

h = subplot(3,4,6);

imhist(img\_noise);

formattedText = {strcat(t\_s,'hist(I\_n)');};

title(h, formattedText);

h = subplot(3,4,7);

binary\_img = img\_noise > 210;

binary\_img = 1 - binary\_img;

imshow(binary\_img);

formattedText = {strcat(t\_s,'B\_n t\_n=210');};

title(h, formattedText);

h = subplot(3,4,8);

[labels, ~] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B\_n');};

title(h, formattedText);

%Noise reduction

h = subplot(3,4,9);

img\_noise\_reduction = gaussian\_smoothing\_filter(img\_noise,3,3,0.53);

imshow(img\_noise\_reduction);

formattedText = {strcat(t\_s,'Filtration - I\_f');};

title(h, formattedText);

h = subplot(3,4,10);

imhist(img\_noise\_reduction);

formattedText = {strcat(t\_s,'hist(I\_f)');};

title(h, formattedText);

h = subplot(3,4,11);

binary\_img = img\_noise\_reduction > 210;

binary\_img = 1 - binary\_img;

imshow(binary\_img);

formattedText = {strcat(t\_s,'B\_f t\_f=210');};

title(h, formattedText);

h = subplot(3,4,12);

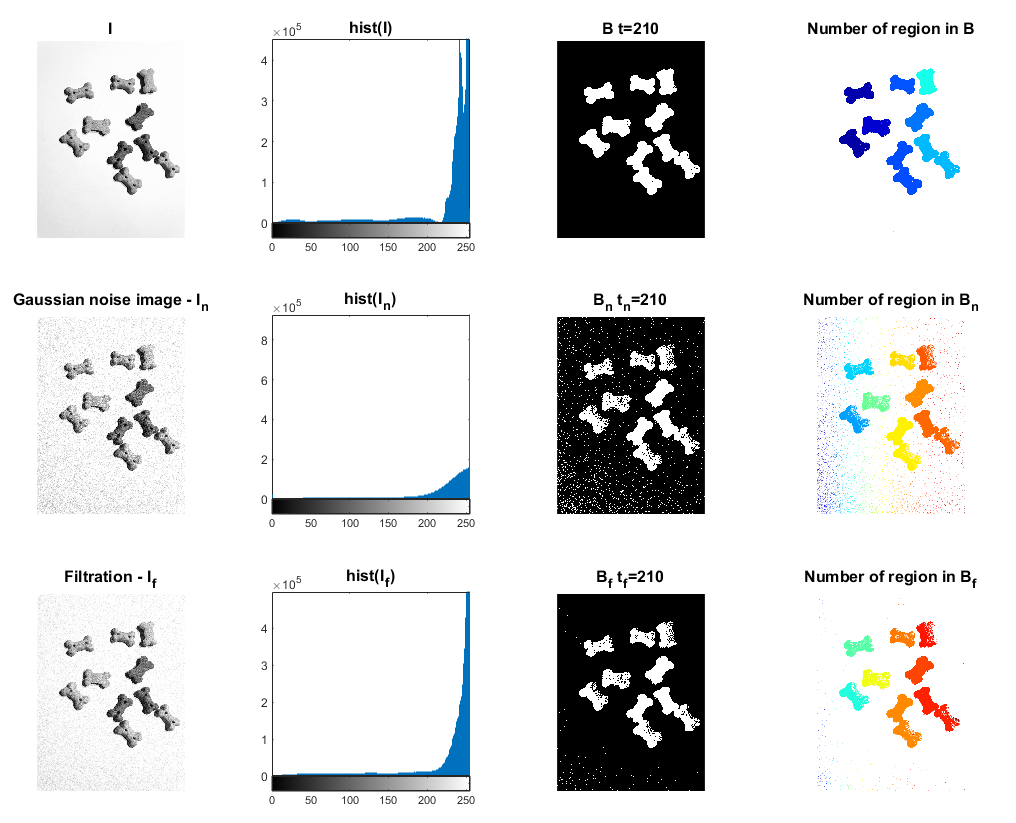
[labels, ~] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B\_f');};

title(h, formattedText);



**Salt & Pepper noise**

figure

% Original image

h = subplot(3,4,1);

imshow(img);

formattedText = {strcat(t\_s,'I');};

title(h, formattedText);

h = subplot(3,4,2);

imhist(img);

formattedText = {strcat(t\_s,'hist(I)');};

title(h, formattedText);

h = subplot(3,4,3);

binary\_img = img > 210;

binary\_img = 1 - binary\_img;

imshow(binary\_img);

formattedText = {strcat(t\_s,'B t=210');};

title(h, formattedText);

h = subplot(3,4,4);

[labels, ~] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B');};

title(h, formattedText);

% Noise

h = subplot(3,4,5);

img\_noise = imnoise(img,'salt & pepper');

imshow(img\_noise);

formattedText = {strcat(t\_s,'Gaussian noise image - I\_n');};

title(h, formattedText);

h = subplot(3,4,6);

imhist(img\_noise);

formattedText = {strcat(t\_s,'hist(I\_n)');};

title(h, formattedText);

h = subplot(3,4,7);

binary\_img = img\_noise > 210;

binary\_img = 1 - binary\_img;

imshow(binary\_img);

formattedText = {strcat(t\_s,'B\_n t\_n=210');};

title(h, formattedText);

h = subplot(3,4,8);

[labels, ~] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B\_n');};

title(h, formattedText);

%Noise reduction

h = subplot(3,4,9);

img\_noise\_reduction = LUM\_filter(img\_noise,3,3,2);

imshow(img\_noise\_reduction);

formattedText = {strcat(t\_s,'Filtration - I\_f');};

title(h, formattedText);

h = subplot(3,4,10);

imhist(img\_noise\_reduction);

formattedText = {strcat(t\_s,'hist(I\_f)');};

title(h, formattedText);

h = subplot(3,4,11);

binary\_img = img\_noise\_reduction > 210;

binary\_img = 1 - binary\_img;

imshow(binary\_img);

formattedText = {strcat(t\_s,'B\_f t\_f=210');};

title(h, formattedText);

h = subplot(3,4,12);

[labels, ~] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

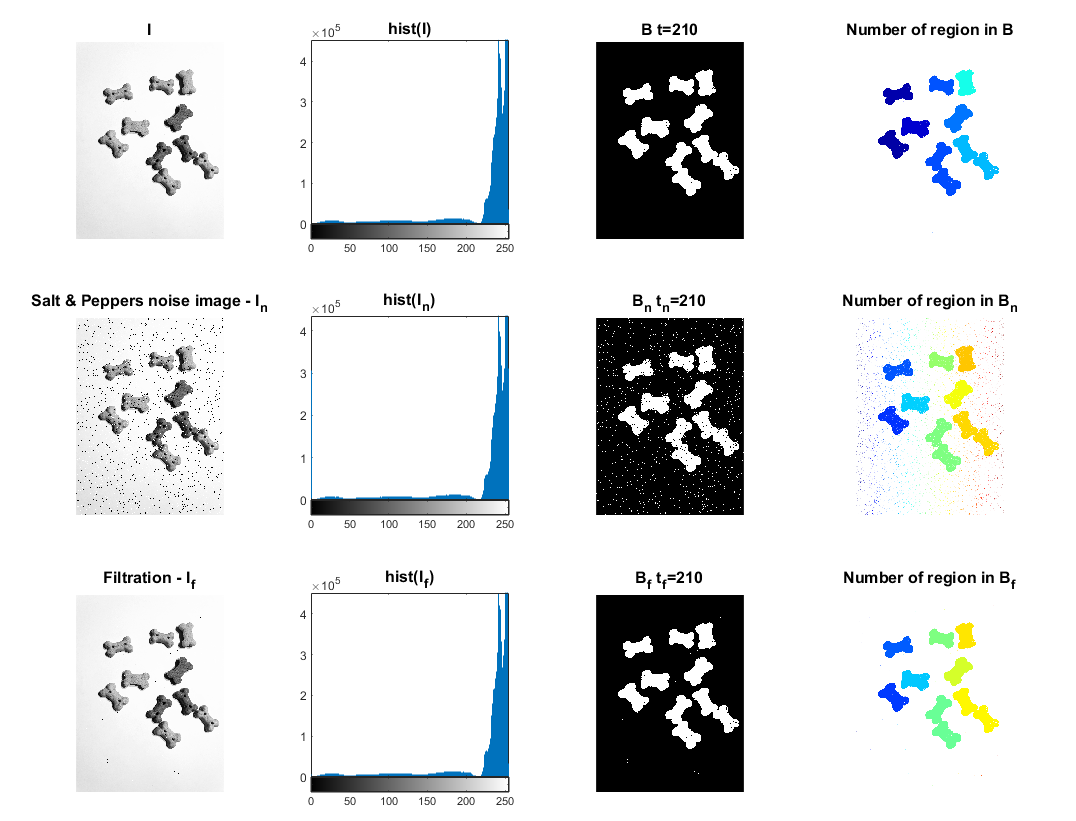
imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B\_f');};

title(h, formattedText);

formattedText = {strcat(ft\_s,'Effect of preprocessing on binarization with Salt & Peppers Noise');};

sgtitle(formattedText);



Both gaussian and salt&peppers noise affect the result. However, the LUM filter for salt&peppers noise and, Gaussian smoothing filter for Gaussian noise, work pretty well.

However, there are some drawbacks and advantages of applying these kinds of filters for binarized images. The advantage of the use is that the contours of the objects became a little bit smoother. Some single pixels, which were earlier wrongly binarized, were corrected. The drawback is that after filtration some small details became changed or removed. It can be seen in some scattered dots on the background, but they can easily be removed using morphological operations such as erosion.

## 4) Postprocessing - morphological operations

subplot(3,5,1),imshow(img)

title('Input image')

subplot(3,5,6)

binary\_img = img > 210;

binary\_img = 1 - binary\_img;

imshow(binary\_img)

title('Binary image')

h = subplot(3,5,11);

[labels, N\_object] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B'); strcat(s\_s,'N =',string(N\_object))};

title(h, formattedText);

%Erosion

subplot(3,5,7)

binary\_img = erosion(binary\_img,5,5);

imshow(binary\_img);

title('Erosion')

h = subplot(3,5,12);

[labels, N\_object] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B'); strcat(s\_s,'N =',string(N\_object))};

title(h, formattedText);

% Dilation

subplot(3,5,8)

binary\_img = dilation(binary\_img,5,5);

imshow(binary\_img)

title('Dilation')

h = subplot(3,5,13);

[labels, N\_object] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B'); strcat(s\_s,'N =',string(N\_object))};

title(h, formattedText);

% Opening

subplot(3,5,9)

for i =1:5

binary\_img = dilation(erosion(binary\_img,5,5),5,5);

end

imshow(binary\_img);

title('Opening')

h = subplot(3,5,14);

[labels, N\_object] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B'); strcat(s\_s,'N =',string(N\_object))};

title(h, formattedText);

% Closing

subplot(3,5,10)

binary\_img = erosion(dilation(binary\_img,7,7),3,3);

imshow(binary\_img);

title('Closing')

h = subplot(3,5,15);

[labels, N\_object] =bwlabel(binary\_img);

labels\_rgb=label2rgb(labels);

imshow(labels\_rgb);

formattedText = {strcat(t\_s,'Number of region in B'); strcat(s\_s,'N =',string(N\_object))};

title(h, formattedText);

## 

## Otsu method function

function [binary\_img, threshold] = otsu\_method(img)

n=imhist(img);

N=sum(n);

max=0;

for i=1:256

P(i)=n(i)/N; %Computing the probability of each intensity level

end

for T=1:255

w0=sum(P(1:T)); % Probability of class 1

w1=sum(P(T+1:256)); %probability of class2

u0=dot([0:T-1],P(1:T))/w0; % class mean u0

u1=dot([T:255],P(T+1:256))/w1; % class mean u1

sigma=w0\*w1\*((u1-u0)^2); % compute variance between class

if sigma>max

max=sigma; % update the value maximum sigma

threshold=T-1; % desired threshold corresponds to maximum variance of between class

end

end

binary\_img = img > threshold;

binary\_img = 1 - binary\_img;

end

## LUM filter function

function output\_img = LUM\_filter(img, a, b, k)

[m ,n] = size(img);

mask = zeros(a,b);

output\_img = img;

start = ceil(a/2);

for x = start:1:m-start

for y = start:1:n-start

for i = 1:1:a

for j = 1:1:b

mask(i,j) = img(x-start+i,y-start+j);

end

end

mask = mask(:)'; %Convert the matrix into an array

mask = sort(mask);

x\_L = mask(ceil(length(mask)/2)-k);

x\_U = mask(ceil(length(mask)/2)+k);

V = [x\_L, x\_U, img(x,y)];

median\_value = median(V,"all");

output\_img(x,y) = median\_value;

mask = zeros(a,b);

end

end

end

## Gaussian smoothing filter

function [output\_img, evaluation] = gaussian\_smoothing\_filter (img\_noise, a, b, sigma)

[m ,n] = size(img\_noise);

output\_img = img\_noise;

mask = zeros(a,b);

for x=1:a

for y=1:b

mask(x,y)=(1/(2\*pi\*sigma^2))\*(exp(-((x-2)^2+(y-2)^2)/(2\*sigma^2)));

end

end

sum=0;

start = ceil(a/2);

for x = start:1:m-start

for y = start:1:n-start

for i = 1:1:a

for j = 1:1:b

sum = sum + (img\_noise(x-start+i,y-start+j)\*mask(i,j));

end

end

output\_img(x,y) = sum;

sum = 0;

end

end

output\_img = uint8(output\_img);

evaluation = psnr(output\_img,img\_noise);

end

## Erosion function

function output\_img = erosion(img, a, b)

[m ,n] = size(img);

pad = zeros(a,b);

output\_img = img;

start = ceil(a/2);

for x = start:1:m-start

for y = start:1:n-start

for i = 1:1:a

for j = 1:1:b

pad(i,j) = img(x-start+i,y-start+j);

end

end

pad = pad(:)'; %Convert the matrix into an array

output\_img(x,y) = min(pad);

pad = zeros(a,b);

end

end

end

## Dilation function

function output\_img = dilation(img, a, b)

[m ,n] = size(img);

pad = zeros(a,b);

output\_img = img;

start = ceil(a/2);

for x = start:1:m-start

for y = start:1:n-start

for i = 1:1:a

for j = 1:1:b

pad(i,j) = img(x-start+i,y-start+j);

end

end

pad = pad(:)'; %Convert the matrix into an array

output\_img(x,y) = max(pad);

pad = zeros(a,b);

end

end

end