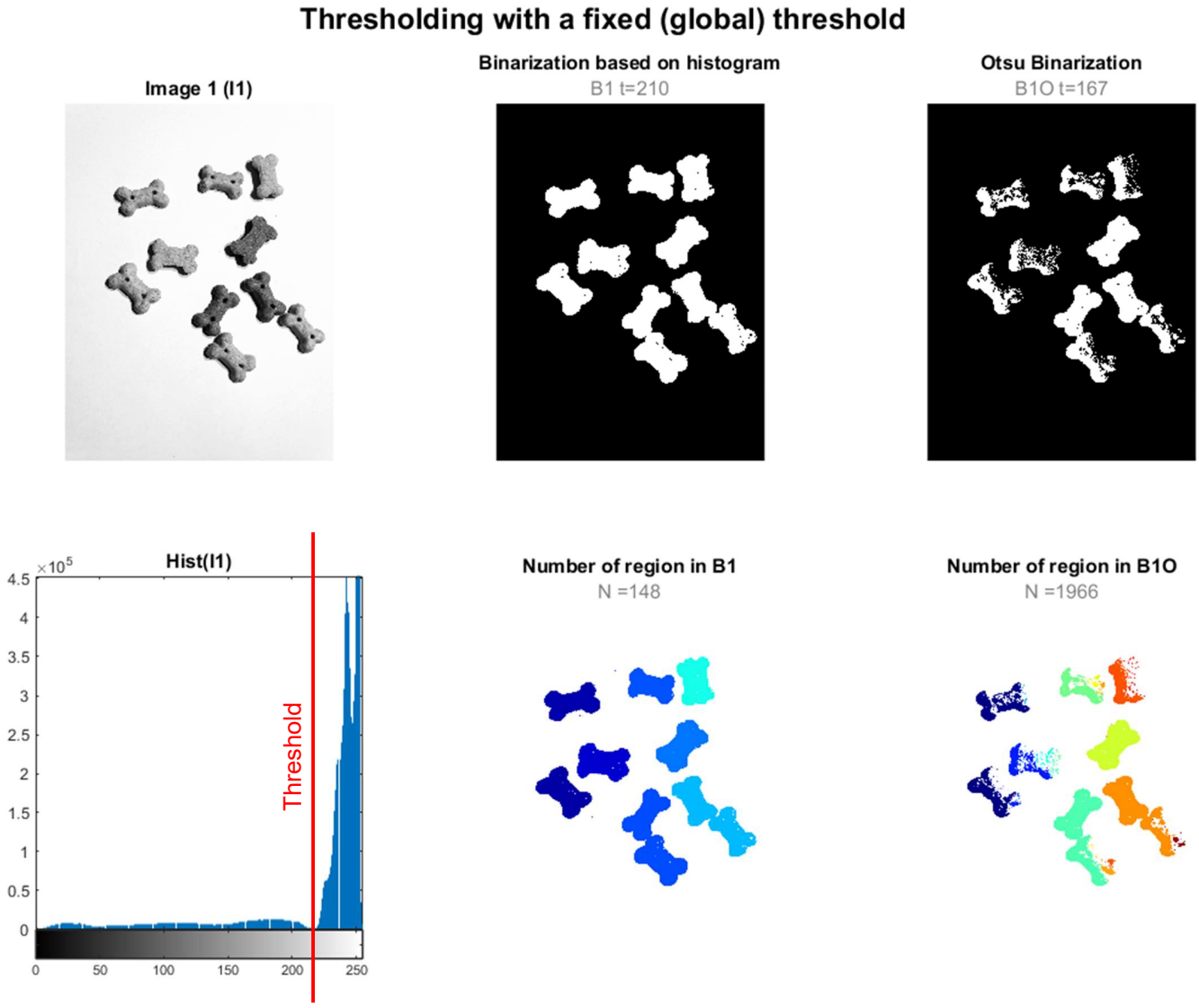
EX5 - Simple Image Segmentation

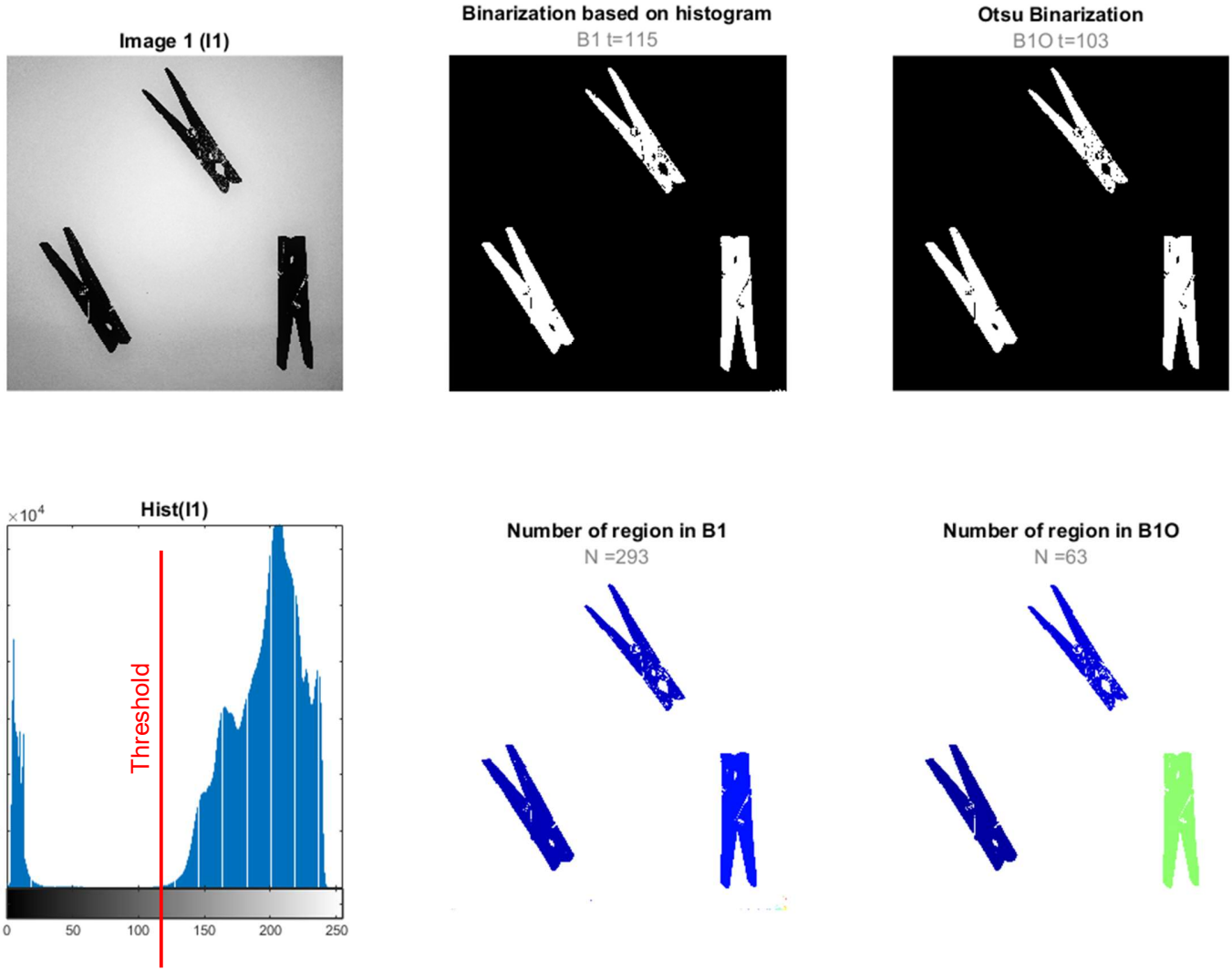
2) Thresholding with a fixed (global) threshold

```
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);
```



Thresholding with a fixed (global) threshold



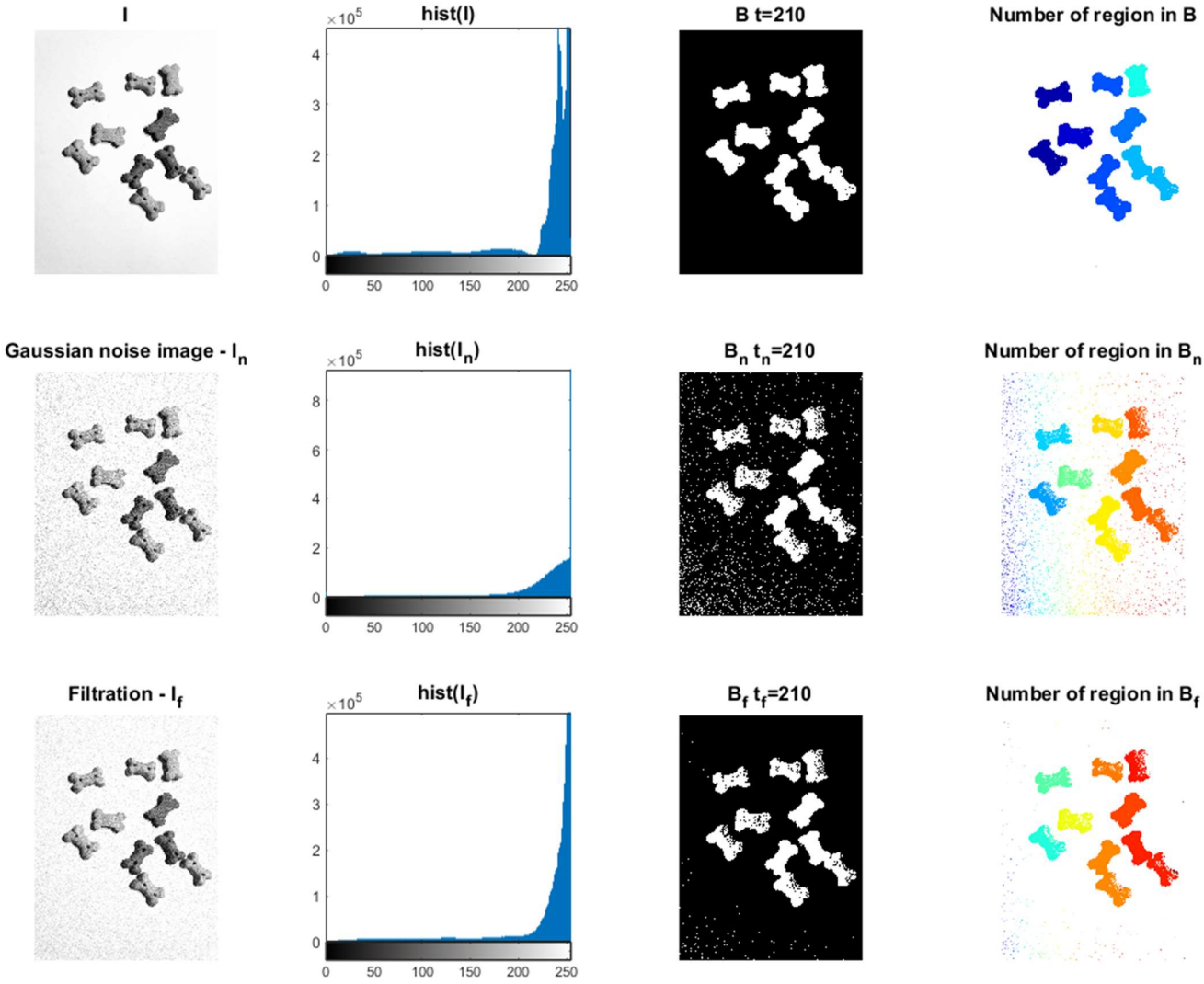
The binarization result are quite good, with interest object well extracted from the background. The presence noise (visible with small dots), contourns not well defined and holes in the object can be easily solved with further morphological operations.

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);
```

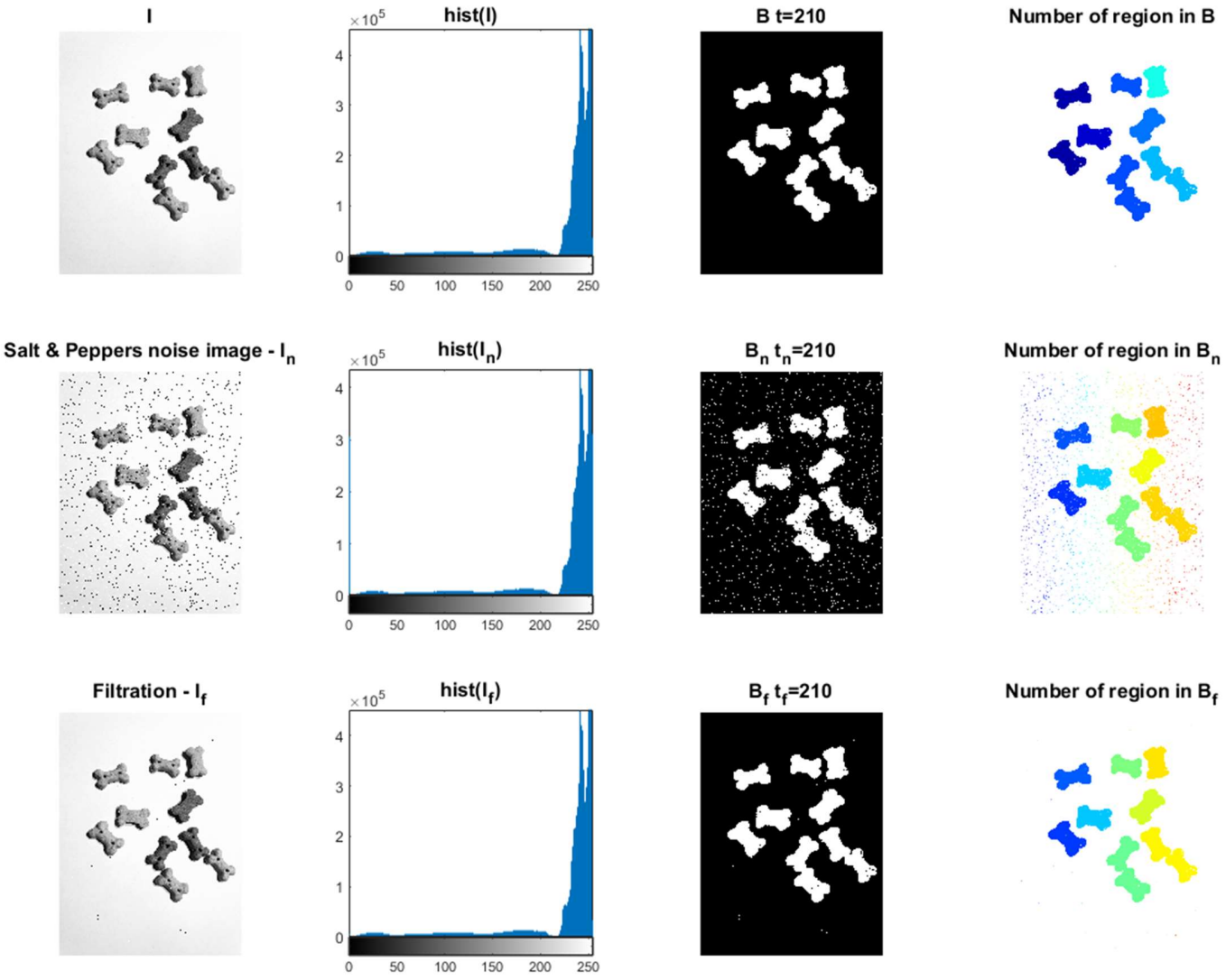


Salt & Pepper noise

```
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);
```

Input image

Binary image

Erosion

Dilation

Opening

Closing

Number of region in B
N =148

Number of region in B
N =39

Number of region in B
N =12

Number of region in B
N =12

Number of region in B
N =9

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
```