Image Enhancement by Point Processing

Created by:

Asaf Kamer

stav sabag

```
clearvars;
clear all ;
close all ;
clc ;
```

Table of Contents

Loading and displaying the image

```
% Specify the filename of the COSAR file
filename = 'IMAGE_HH_SRA_spot_048.cos';

% Call the readCosFile function to read the data and information from the file
[HH_image, info] = readCosFile(filename,1);


% Display some basic information about the data and the file
disp(['File ' filename ' contains ' num2str(size(HH_image,2)) ' range lines and ' ...
    num2str(size(HH_image,3)) ' azimuth lines.'])
```

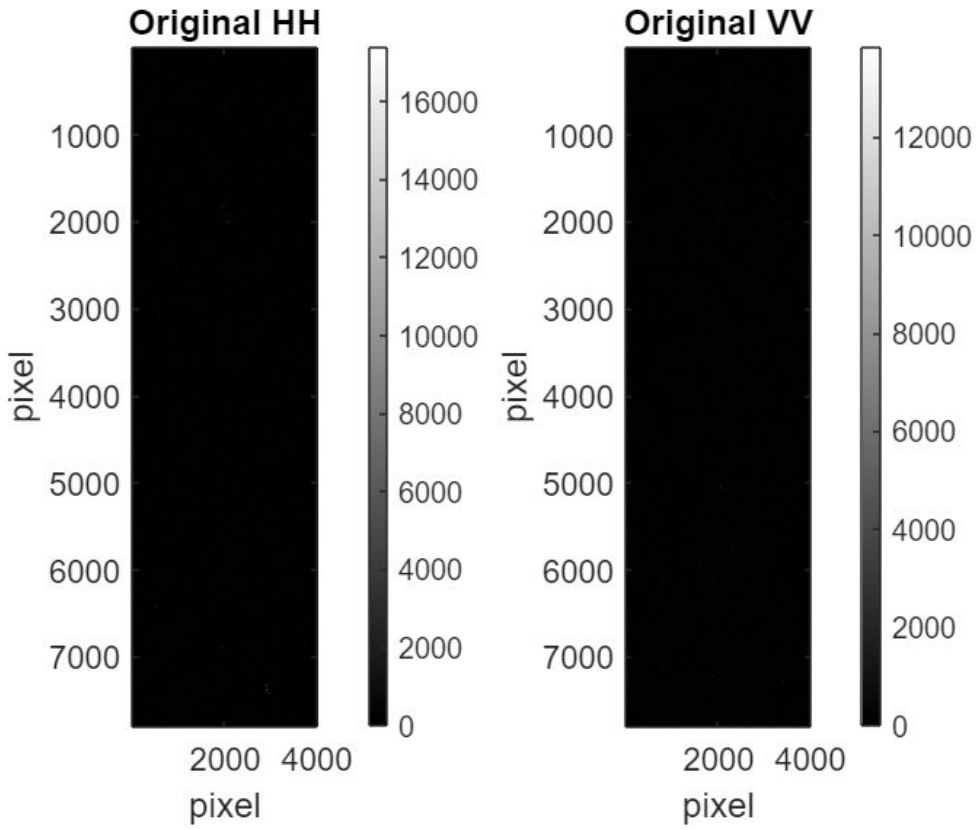
```
File IMAGE_HH_SRA_spot_048.cos contains 7808 range lines and 4018 azimuth lines.
```

```
% Specify the filename of the COSAR file
filename = 'IMAGE_VV_SRA_spot_048.cos';

% Call the readCosFile function to read the data and information from the file
[VV_image, info] = readCosFile(filename,1);
```

```
% Display some basic information about the data and the file
disp(['File ' filename ' contains ' num2str(size(VV_image,2)) ' range lines and ' ...
    num2str(size(VV_image,3)) ' azimuth lines.'])
```

```
File IMAGE_VV_SRA_spot_048.cos contains 7808 range lines and 4018 azimuth lines.
```

```
% Combine the real and imaginary parts of the first image and the second into a single complex
abs_HH_image = (double(HH_image(1,:,:)).^2 + double(HH_image(2,:,:)).^2).^0.5 ;
abs_HH_image = uint16(squeeze(abs_HH_image)) ;
% Compute the magnitude of each complex number in the matrix


abs_VV_image = (double(VV_image(1,:,:)).^2 + double(VV_image(2,:,:)).^2).^0.5 ;
abs_VV_image = uint16(squeeze(abs_VV_image)) ;

tle1 = 'Original HH' ;
tle2 = 'Original VV' ;
figure ;
subplot(1,2,1)
imagesc(abs_HH_image)
colormap("gray");
colorbar;
title(tle1,'FontSize',18); xlabel("pixel", "FontSize",14); ylabel("pixel", "FontSize",14); set(
subplot(1,2,2)
imagesc(abs_VV_image)
colormap("gray");
colorbar;
title(tle2,'FontSize',18); xlabel("pixel", "FontSize",14); ylabel("pixel", "FontSize",14); set(
```



information gathering

statistical and other parameters

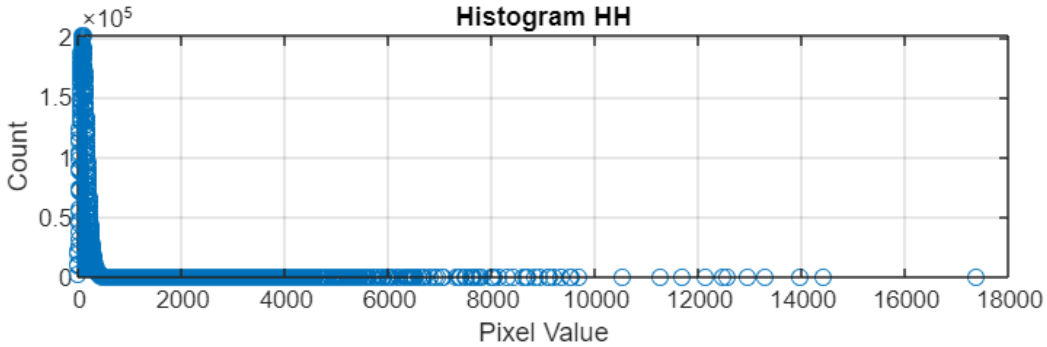
```
% Transformation to a variable of type double
abs_HH_image = double(abs_HH_image) ;
abs_VV_image = double(abs_VV_image) ;

% Compute the average value of the pixels
avg_HH_image = mean(abs_HH_image(:));
avg_VV_image = mean(abs_VV_image(:));

% Compute the standard deviation of the pixels
std_HH_image = std(abs_HH_image(:));
std_VV_image = std(abs_VV_image(:));

% Compute the maximum value of the pixels
max_HH_image = max(abs_HH_image(:));
max_VV_image = max(abs_VV_image(:));

% Compute the minimum value of the pixels
min_HH_image = min(abs_HH_image(:));
min_VV_image = min(abs_VV_image(:));

% Compute the number of pixels
```

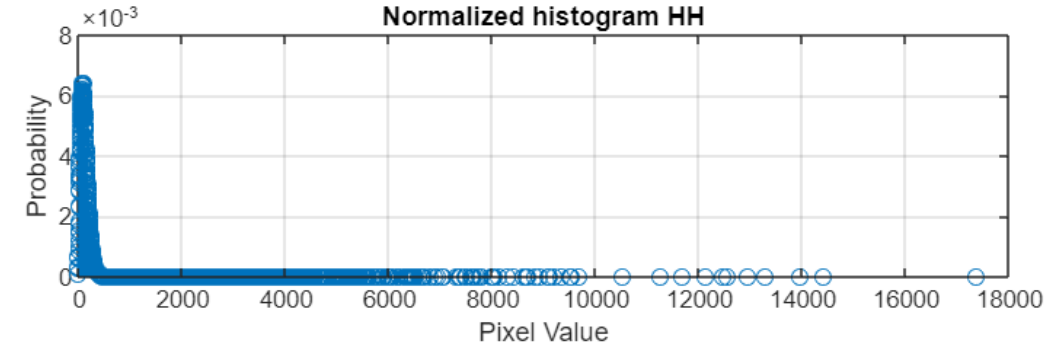
```
num_pixels_HH_image = numel(abs_HH_image);
num_pixels_VV_image = numel(abs_VV_image);

% Transformation to a variable of type uint16
abs_HH_image = uint16(abs_HH_image) ;
abs_VV_image = uint16(abs_VV_image) ;

% Create the table
image_stats_table = table([1;2], [avg_HH_image; avg_VV_image], [std_HH_image; std_VV_image], ..
                          [max_HH_image; max_VV_image], [min_HH_image; min_VV_image], [num_pixe
                          'VariableNames', {'Image Number', 'Average Value', 'Standard Deviatio
                                            'Maximum Value', 'Minimum Value', 'Number of pixels

% Display the table
disp(image_stats_table)
```

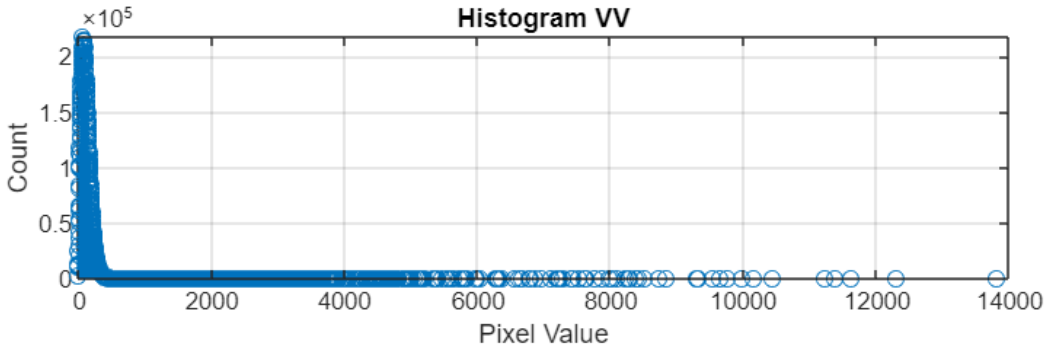
Image Number	Average Value	Standard Deviation	Maximum Value	Minimum Value	Number of pixels
1	124.16	82.86	17372	0	3.1373e+07
2	116.37	75.859	13823	0	3.1373e+07

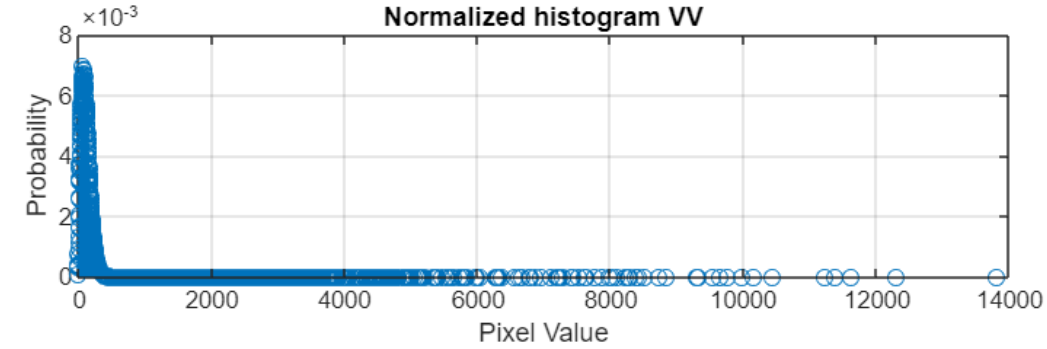
Histogram for the original images

```
% histogram calculation
hist_HH_img = Histogram(abs_HH_image,num_pixels_HH_image) ;
hist_VV_img = Histogram(abs_VV_image,num_pixels_VV_image);

% plot
figure('Position', [0 0 600 800]);
subplot(4,1,1)
stem(hist_HH_img(:,1),hist_HH_img(:,2))
grid on ; xlabel('Pixel Value'); ylabel('Count'); title('Histogram HH');
subplot(4,1,2)
stem(hist_HH_img(:,1),(hist_HH_img(:,2))./num_pixels_HH_image)
grid on ; xlabel('Pixel Value'); ylabel('Probability'); title('Normalized histogram HH');
subplot(4,1,3)
stem(hist_VV_img(:,1),hist_VV_img(:,2))
grid on ; xlabel('Pixel Value'); ylabel('Count'); title('Histogram VV');
subplot(4,1,4)
stem(hist_VV_img(:,1),(hist_VV_img(:,2))./num_pixels_VV_image)
grid on ; xlabel('Pixel Value'); ylabel('Probability'); title('Normalized histogram VV');
```







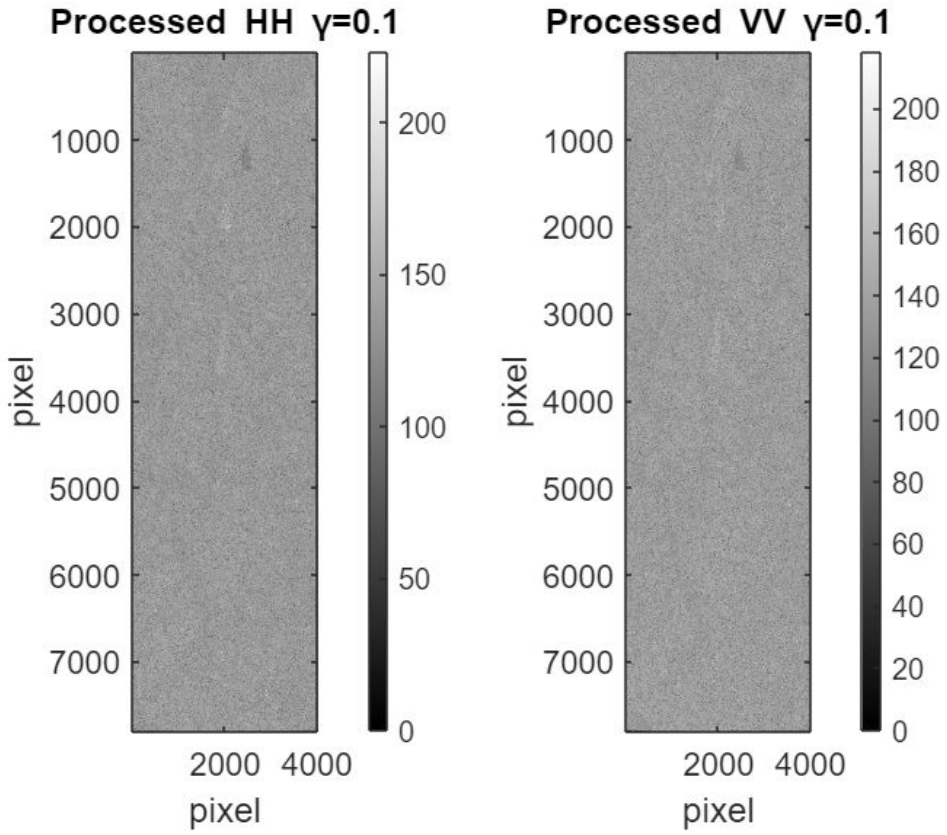


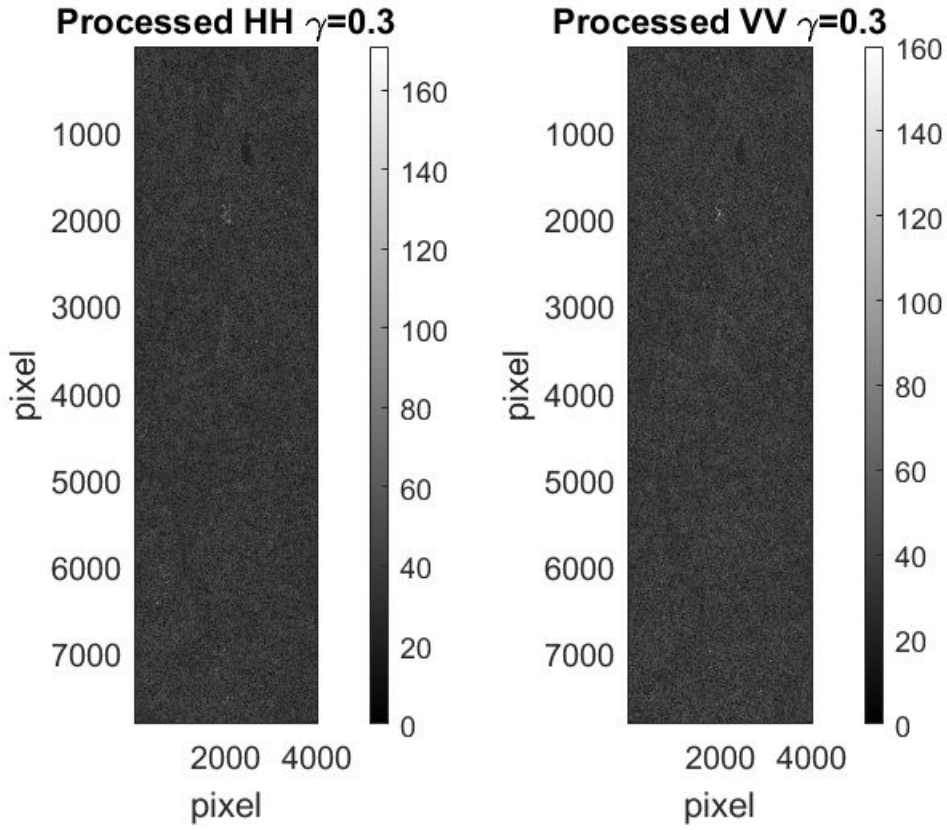
Pixel processing

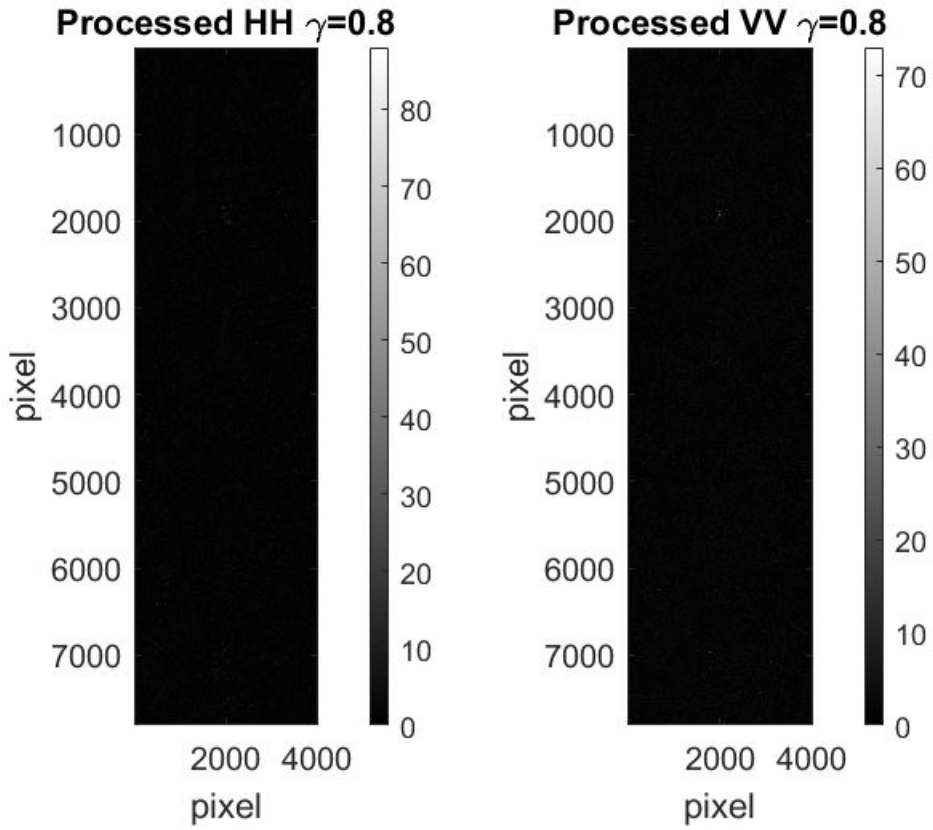
Power-law

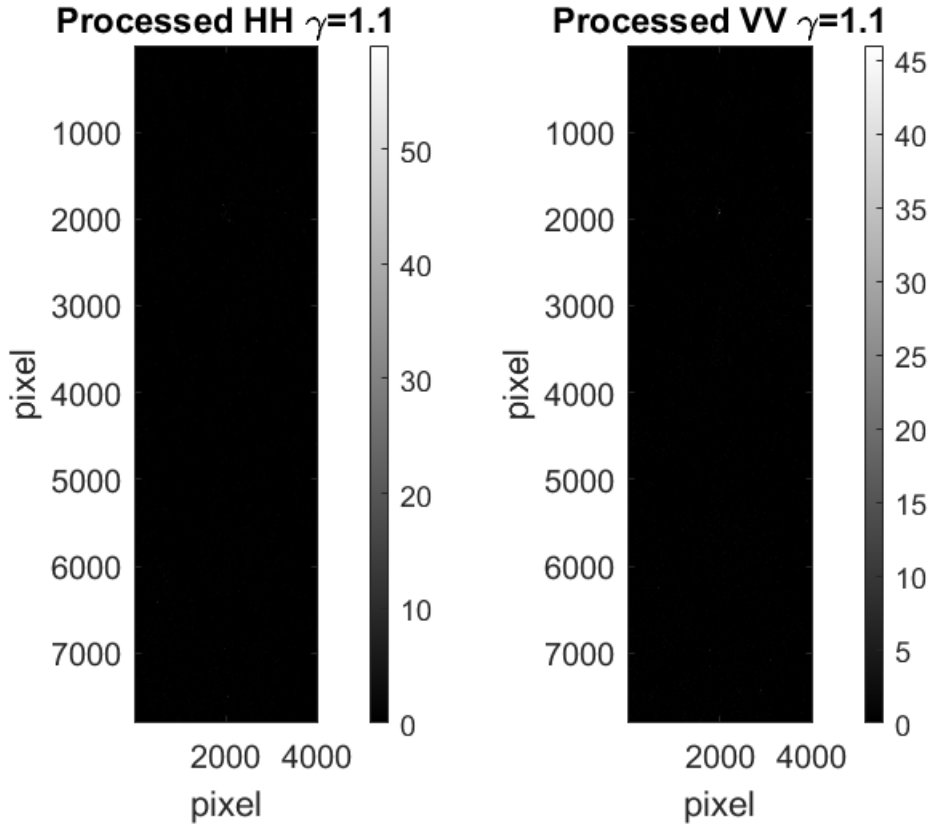
```
% Transformation to a variable of type double
```

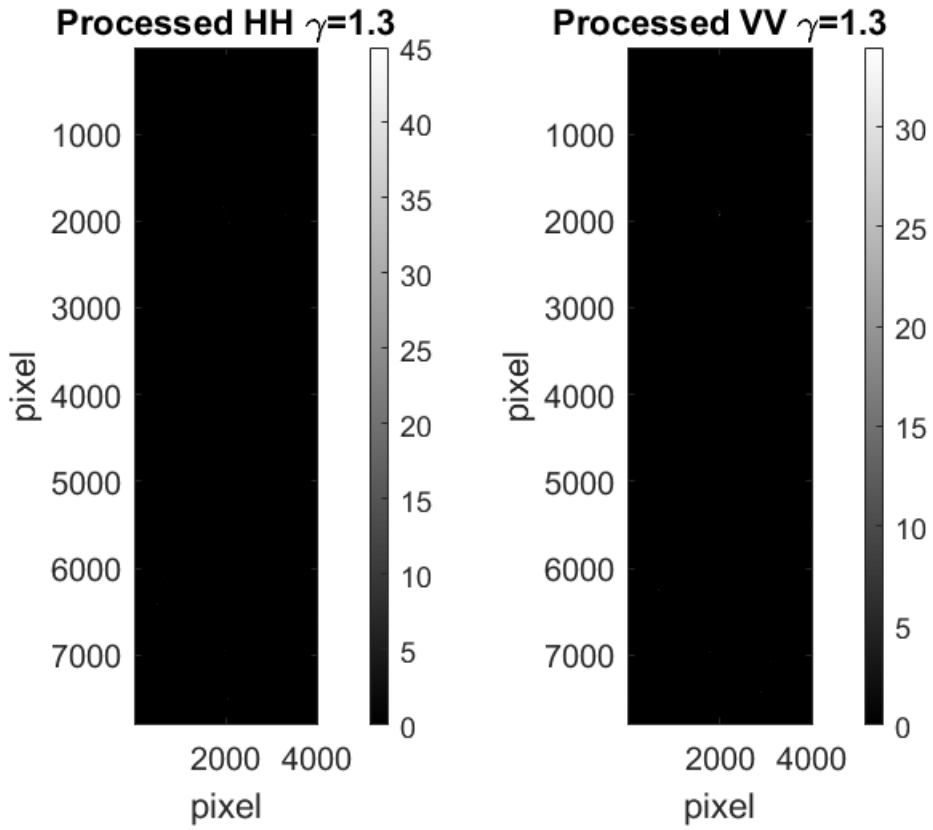
```
abs_HH_image = double(abs_HH_image) ;
abs_VV_image = double(abs_VV_image) ;

%Pixel value transform
% Define gamma values
gamma = [0.1, 0.3, 0.8, 1.1, 1.3];

% Define image transformation parameters
a = [1/(2^16-1), 1/(2^16-1)];
b = [0, 0];

% Process images for each gamma value
for i = 1 : length(gamma)
    % Process first image
    Process_HH_Img = a(1).*(abs_HH_image) + b(1);
    Process_HH_Img = uint8(255.*(Process_HH_Img).^gamma(i));

    % Process second image
    Process_VV_Img = a(2).*(abs_VV_image) + b(2);
    Process_VV_Img = uint8(255.*(Process_VV_Img).^gamma(i));

    % Save processed images in cell array
    processed_images{i, 1} = Process_HH_Img;
    processed_images{i, 2} = Process_VV_Img;

    tle1 = ['Processed HH \gamma=' num2str(gamma(i))];
    tle2 = ['Processed VV \gamma=' num2str(gamma(i))];
    figure ;
    subplot(1,2,1)
    imagesc(Process_HH_Img)
    colormap("gray");
    colorbar;
    title(tle1,'FontSize',18); xlabel("pixel", "FontSize",14); ylabel("pixel", "FontSize",14);
    subplot(1,2,2)
    imagesc(Process_VV_Img)
    colormap("gray");
    colorbar;
    title(tle2,'FontSize',18); xlabel("pixel", "FontSize",14); ylabel("pixel", "FontSize",14);

end
```


Processed HH γ=0.1
Processed VV γ=0.1
pixel
1000
2000
3000
4000
5000
6000
7000
2000 4000
pixel
200
150
100
50
0
200
180
160
140
120
100
80
60
40
20
0


Processed HH γ=0.3
Processed VV γ=0.3
pixel
1000
2000
3000
4000
5000
6000
7000
2000 4000
pixel
160
140
120
100
80
60
40
20
0
160
140
120
100
80
60
40
20
0


Processed HH γ=0.8
Processed VV γ=0.8
pixel
1000
2000
3000
4000
5000
6000
7000
2000 4000
pixel
0
10
20
30
40
50
60
70
80


Processed HH γ=1.1
Processed VV γ=1.1
pixel
1000
2000
3000
4000
5000
6000
7000
2000 4000
pixel
0
5
10
15
20
25
30
35
40
45
50

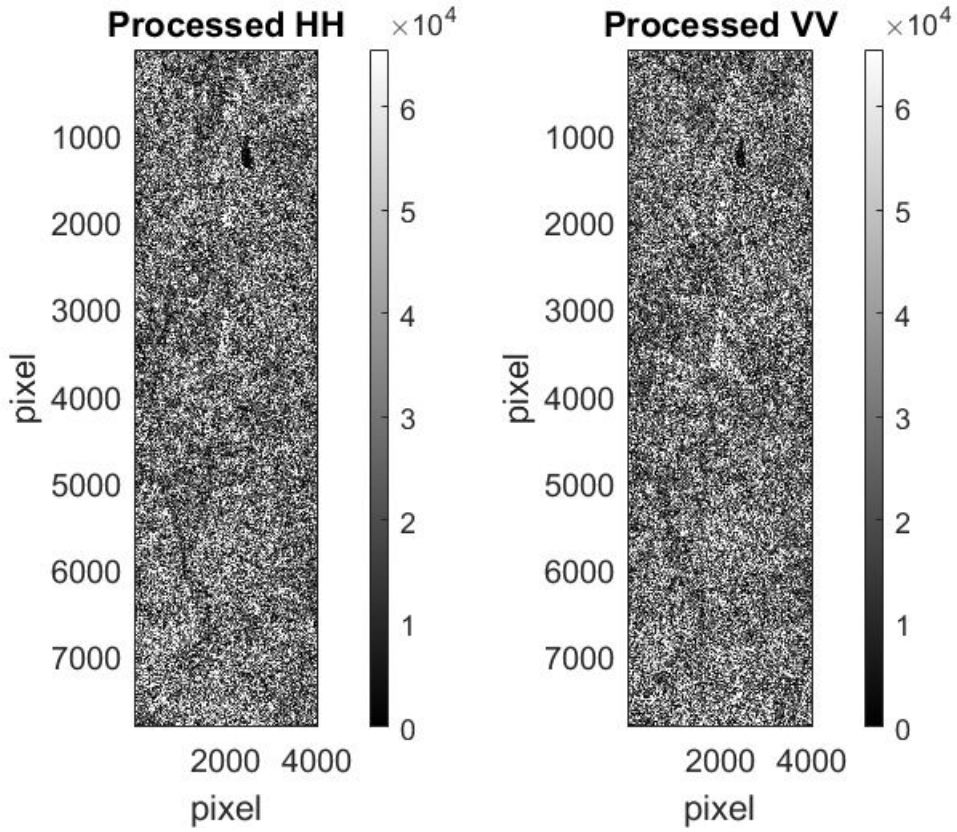



```
% Transformation to a variable of type uint16
abs_HH_image = uint16(abs_HH_image) ;
abs_VV_image = uint16(abs_VV_image) ;
```

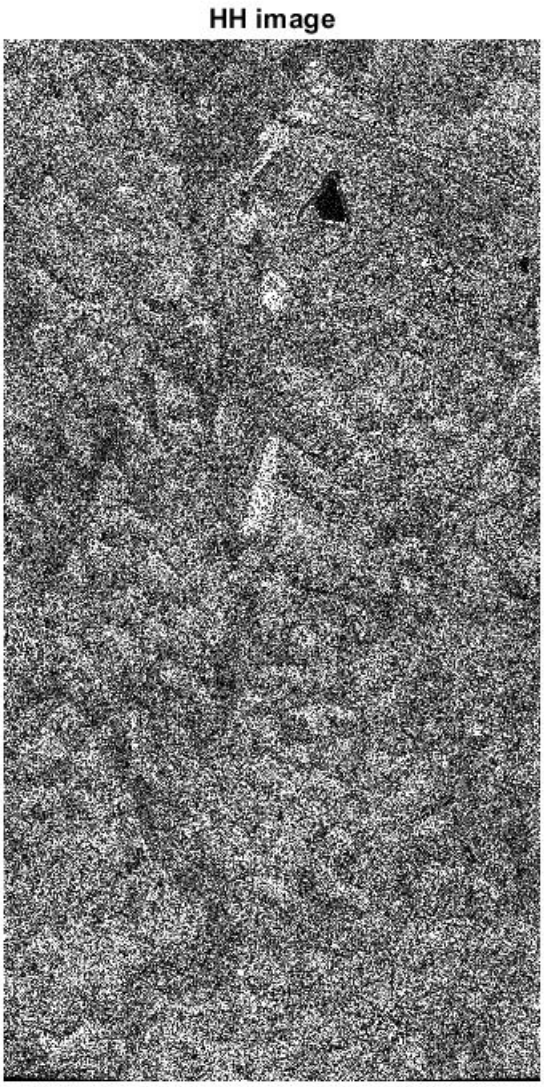
Transformation on light and dark pixels

```
% Transformation to a variable of type double
abs_HH_image = double(abs_HH_image) ;
abs_VV_image = double(abs_VV_image) ;

L = 2^16;

Process_HH_Img = processed_image(abs_HH_image,L) ;
Process_VV_Img = processed_image(abs_VV_image,L) ;

% Transformation to a variable of type uint16
abs_HH_image = uint16(abs_HH_image) ;
abs_VV_image = uint16(abs_VV_image) ;

tle1 = 'Processed HH' ;
tle2 = 'Processed VV' ;
figure ;
subplot(1,2,1)
imagesc(Process_HH_Img)
colormap("gray");
colorbar;
```

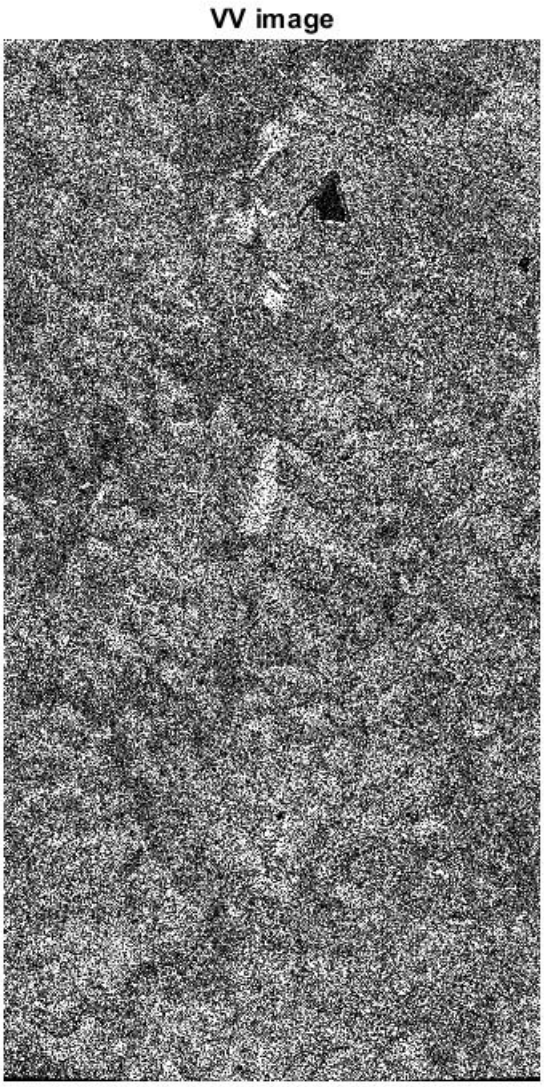
```
title(tle1,'FontSize',18); xlabel("pixel", "FontSize",14); ylabel("pixel", "FontSize",14); set(
subplot(1,2,2)
imagesc(Process_VV_Img)
colormap("gray");
colorbar;
title(tle2,'FontSize',18); xlabel("pixel", "FontSize",14); ylabel("pixel", "FontSize",14); set(
```




```
figure ;
imshow(Process_HH_Img)
title('HH image') ;
```



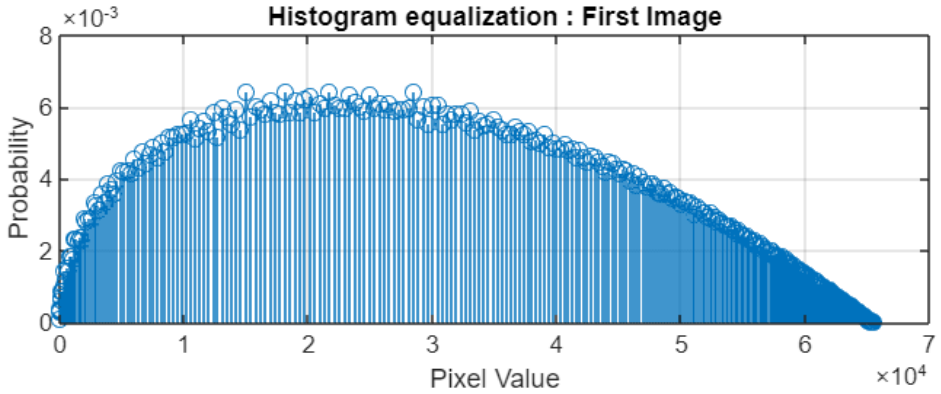

```
figure ;
imshow(Process_VV_Img)
title('VV image') ;
```

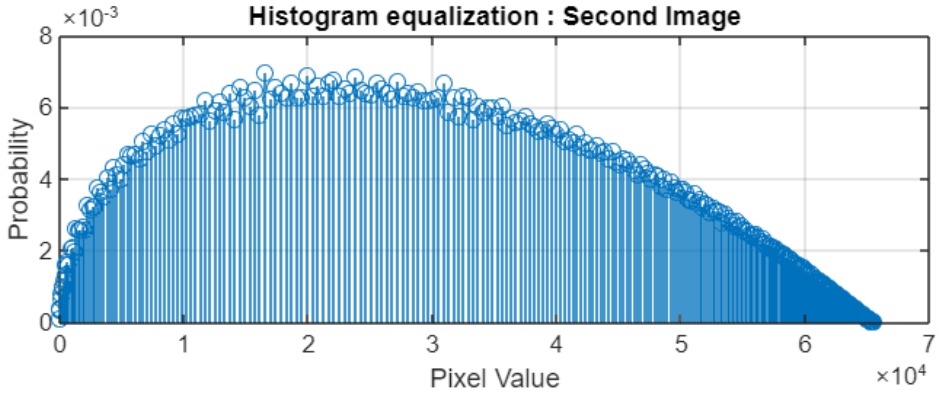


Histogram equalization

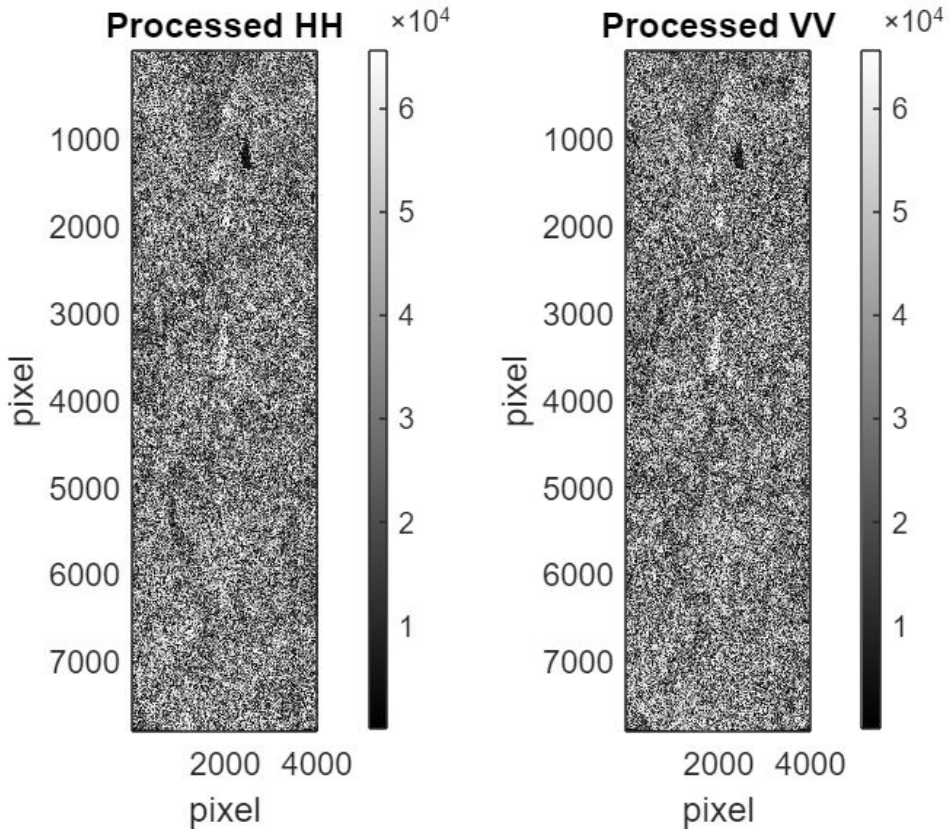
```
% First image :

% CDF calculation
cdf_HH_image = cumsum(hist_HH_img(:,2)./num_pixels_HH_image) ;

% Finding the new values
Histogram_equalization1 = round((L-1) .* cdf_HH_image) ;

% Transformation
Process_HH_Img = abs_HH_image ;
for i = 1 : length(Histogram_equalization1)
    Process_HH_Img(find(abs_HH_image == hist_HH_img(i,1))) = Histogram_equalization1(i) ;
end
```

```
% Histogram equalization display.


% Counting the amount of values
hist_HH_img_Eq = Histogram(Process_HH_Img,num_pixels_HH_image) ;


% Second image :

% CDF calculation
cdf_VV_image = cumsum(hist_VV_img(:,2)./num_pixels_VV_image) ;

% Finding the new values
Histogram_equalization2 = round((L-1) .* cdf_VV_image) ;

% Transformation
Process_VV_Img = abs_VV_image ;
for i = 1 : length(Histogram_equalization2)
    Process_VV_Img(find(abs_VV_image == hist_VV_img(i,1))) = Histogram_equalization2(i) ;
end

% Histogram equalization display.

hist_VV_img_Eq = Histogram(Process_VV_Img,num_pixels_VV_image) ;

% plot

figure ;
subplot(2,1,1) ;
bins = length(hist_HH_img_Eq) ;
stem(hist_HH_img_Eq(1:bins,1),(hist_HH_img_Eq(1:bins,2))./num_pixels_HH_image)
grid on ; xlabel('Pixel Value'); ylabel('Probability'); title('Histogram equalization : First I
subplot(2,1,2) ;
bins = length(hist_VV_img_Eq) ;
stem(hist_VV_img_Eq(1:bins,1),(hist_VV_img_Eq(1:bins,2))./num_pixels_VV_image)
grid on ; xlabel('Pixel Value'); ylabel('Probability'); title('Histogram equalization : Second
```

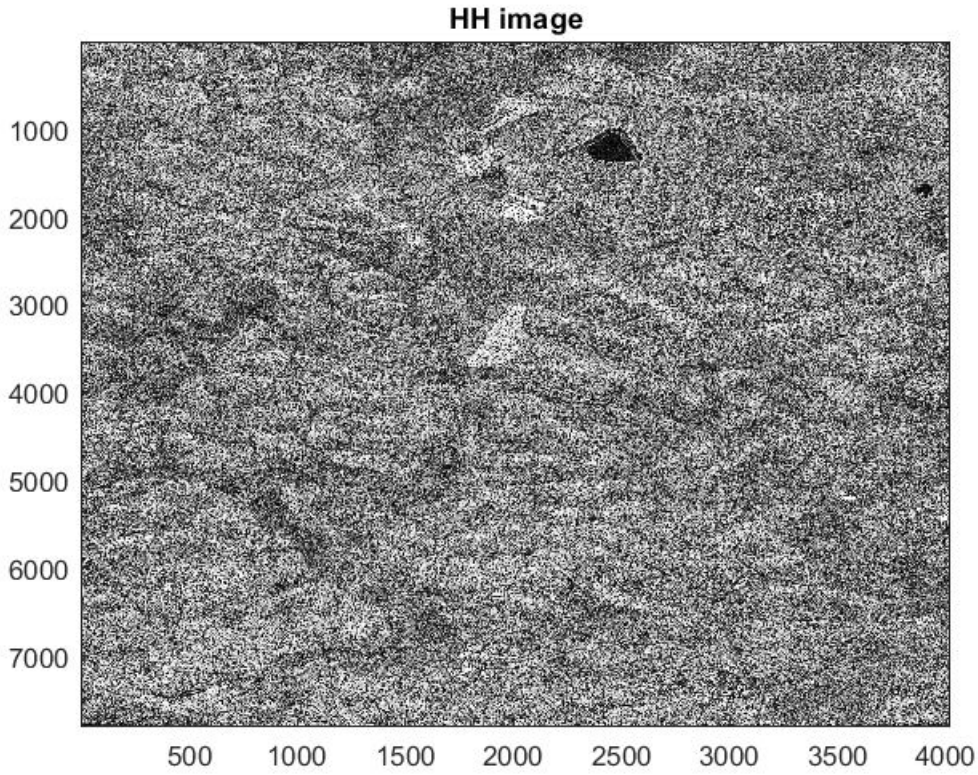




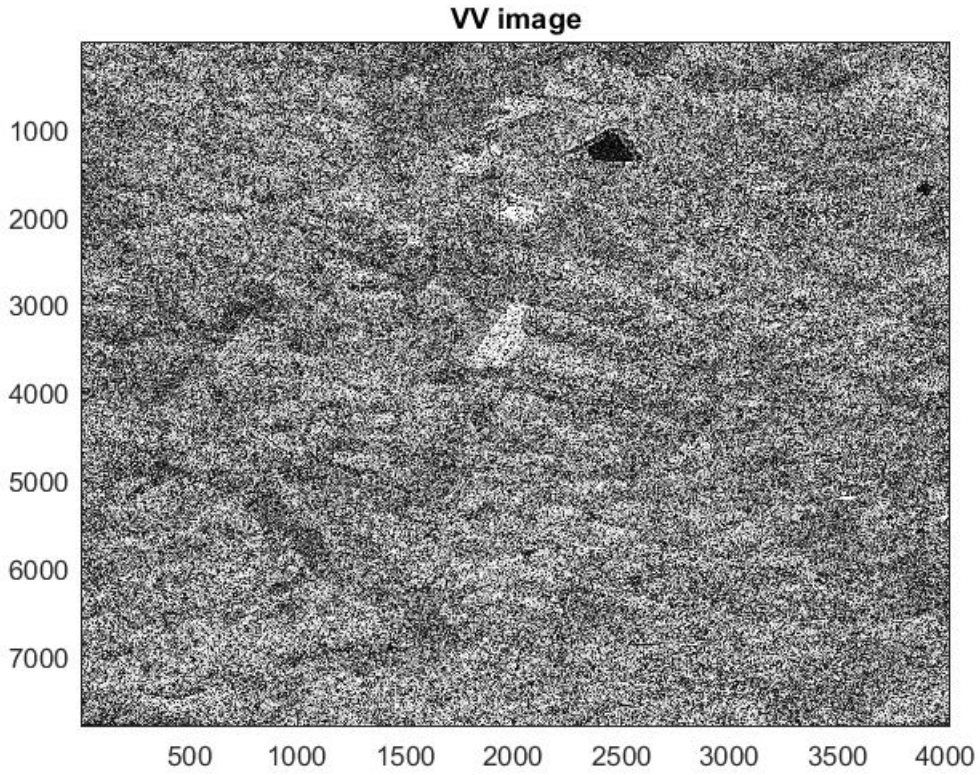

```
tle1 = 'Processed HH' ;
tle2 = 'Processed VV' ;
figure ;
subplot(1,2,1)
imagesc(Process_HH_Img)
colormap("gray");
colorbar;
title(tle1,'FontSize',18); xlabel("pixel", "FontSize",14); ylabel("pixel", "FontSize",14); set(
subplot(1,2,2)
imagesc(Process_VV_Img)
colormap("gray");
colorbar;
title(tle2,'FontSize',18); xlabel("pixel", "FontSize",14); ylabel("pixel", "FontSize",14); set(
```




```
figure ;
imagesc(Process_HH_Img)
colormap("gray");
title('HH image') ;
```



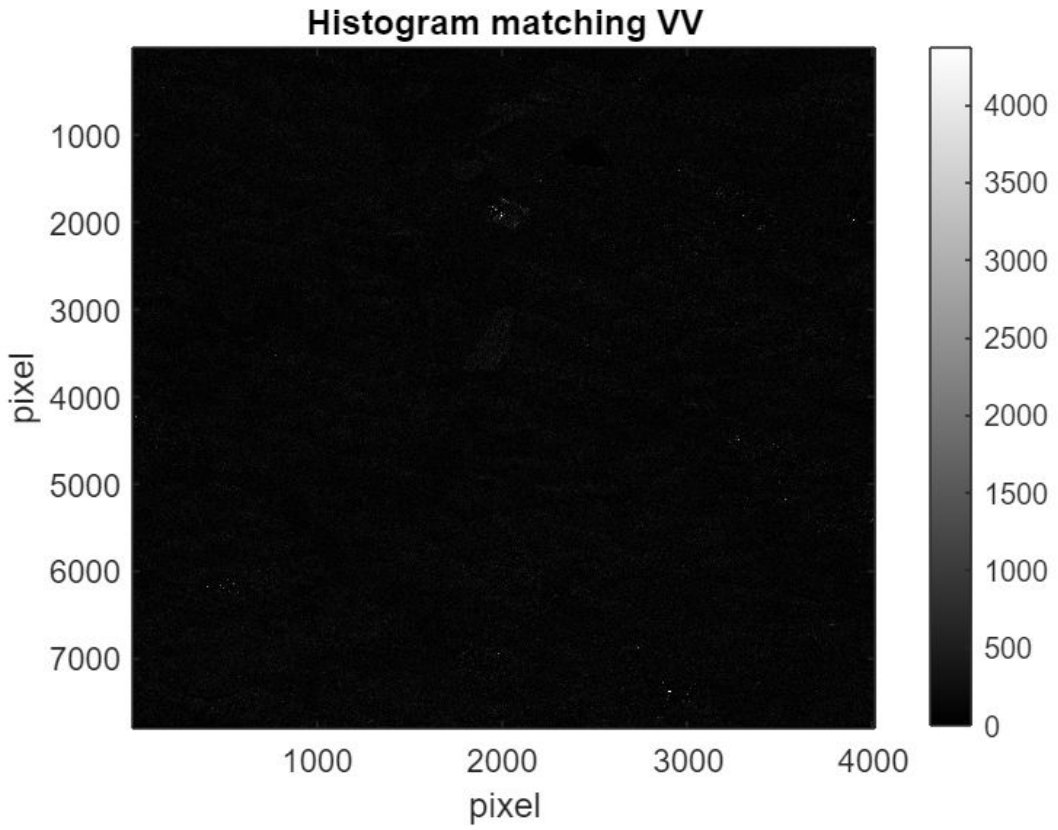

```
figure ;
imagesc(Process_VV_Img)
colormap("gray");
title('VV image') ;
```



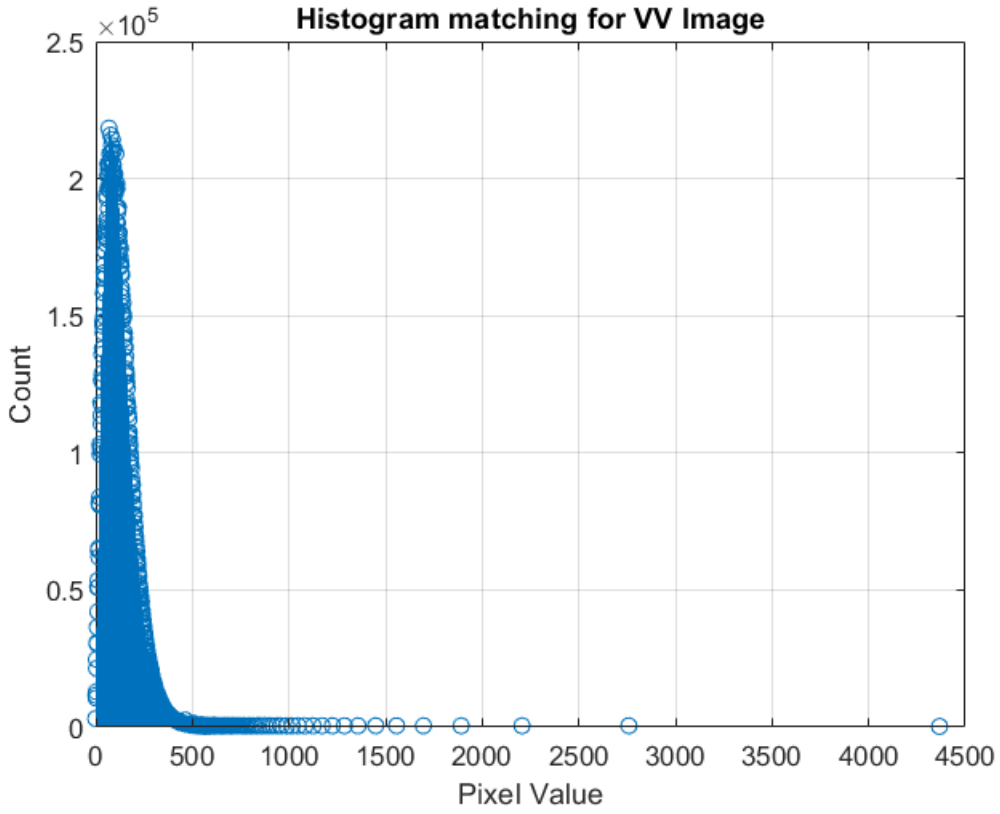
Histogram Matching :

In this section, we will perform a histogram matching on the VV image based on the reference provided by the HH image.

```
% Transformation to a variable of type double
abs_HH_image = double(abs_HH_image) ;
abs_VV_image = double(abs_VV_image) ;

Process_VV_Img = abs_VV_image ;

for i=1:length(Histogram_equalization2)
    % Finding the inverse transformation
    min_diff = min(abs(Histogram_equalization2(i) - Histogram_equalization1)) ;
    idx = min(find(abs(Histogram_equalization2(i) - Histogram_equalization1) == min_diff));
    source_value = hist_VV_img(i,1) ;
    target_value = hist_HH_img(idx,1) ;
    Process_VV_Img(find(abs_VV_image == hist_VV_img(i,1))) = target_value ;
end

figure ;
imagesc(Process_VV_Img)
colormap("gray");
colorbar;
title('Histogram matching VV','FontSize',18); xlabel("pixel", "FontSize",14); ylabel("pixel", '
```




```
Histogram_matching_VV = Histogram(Process_VV_Img,num_pixels_VV_image) ;

% Transformation to a variable of type uint16
abs_HH_image = uint16(abs_HH_image) ;
abs_VV_image = uint16(abs_VV_image) ;

figure ;
bins = length(Histogram_matching_VV) ;
stem(Histogram_matching_VV(1:bins,1),(Histogram_matching_VV(1:bins,2)))
grid on ; xlabel('Pixel Value'); ylabel('Count'); title('Histogram matching for VV Image');
```



Function

Histogram

```
function unique_vals_Img = Histogram(image,num_pixels)
    % Find the unique values
    unique_vals_Img = double(unique(image));

    % Counting the amount of values
    k = reshape(image,1,num_pixels) ;
    k = sort(k) ;
    unique_vals_Img(:,2) = (groupcounts(k'))' ;
end
```

processed image

```
function ProcessImg = processed_image(image,L)

    % Arrange the pixels in ascending order
    sorted_vals = sort(image(:));

    % Finding the threshold values: the 90% value and the 5% value
    threshold_val_90 = sorted_vals(round(0.9*numel(sorted_vals)));
```

```
    threshold_val_5 = sorted_vals(round(0.05*numel(sorted_vals)));

    % Transformation :
    ProcessImg = image ;

    %Transformation to all values greater than the 90% value
    ProcessImg(find(ProcessImg>threshold_val_90)) = threshold_val_90 ;

    %Transformation to all values smaller than 5%
    ProcessImg(find(ProcessImg<threshold_val_5)) = threshold_val_5 ;

    % Transformation: the range of pixels between 0 and L-1 :

    % Transformation: values between 0 and 1
    ProcessImg = ProcessImg - min(min(ProcessImg)) ;
    ProcessImg = (ProcessImg)./double(max(max(ProcessImg))) ;

    % Maximum value L-1
    ProcessImg = (L-1).* ProcessImg ;
    ProcessImg = uint16(ProcessImg) ;
end
```