ML assisted Despeckling of SAR Images

Rakshith N Madapura PES1UG20EC152

Rohan M Rao PES PES1UG20EC156

Shreyas Golasangi PES1UG20EC186

Objectives:

- Train a Decision Tree Regressor model to predict the appropriate kernel size of the Lee filter.
- After de-speckling, post-process the image and apply 10 layer pseudo coloring for better visualization.
- Compare our output with that of other filters.

Introduction / Theoretical Details:

What is a SAR image?

- SAR image is produced by processing received EM waves (usually Radio) from the reflecting surfaces.
- The resulting image is built up from the strength and time delay of the received signal.

What is speckling?

- Speckle is an undesirable granular noise on an image.
- It is caused by the interaction of the out of phase waves reflected from the target.

Lee Filter:

- Uses spatial statistics (coefficient of variation) within individual filter windows. Each pixel is put into one of three classes, which are treated as follows:

  - **Homogeneous**: The pixel value is replaced by the average of the filter window.

- **Heterogeneous**: The pixel value is replaced by a weighted average.
- **Point target**: The pixel value is not changed.

$$Y_{ij} = \overline{K} + W * (C - \overline{K})$$

Where

Y_{ij} is the despeckled image

$\overline{K}$ is the mean of the kernel/window

W is the weighing function

C is the center element in the kernel/window

To calculate W:

$$W = \frac{\sigma_k^2}{(\sigma_k^2 + \sigma^2)}$$

Where

σ^2 is the variance of the reference image

σ_k^2 is the variance of the pixels in the kernel/window of the speckled image

<u>What are we trying to predict through our decision tree regressor model?</u>

- ***Kernel size***.

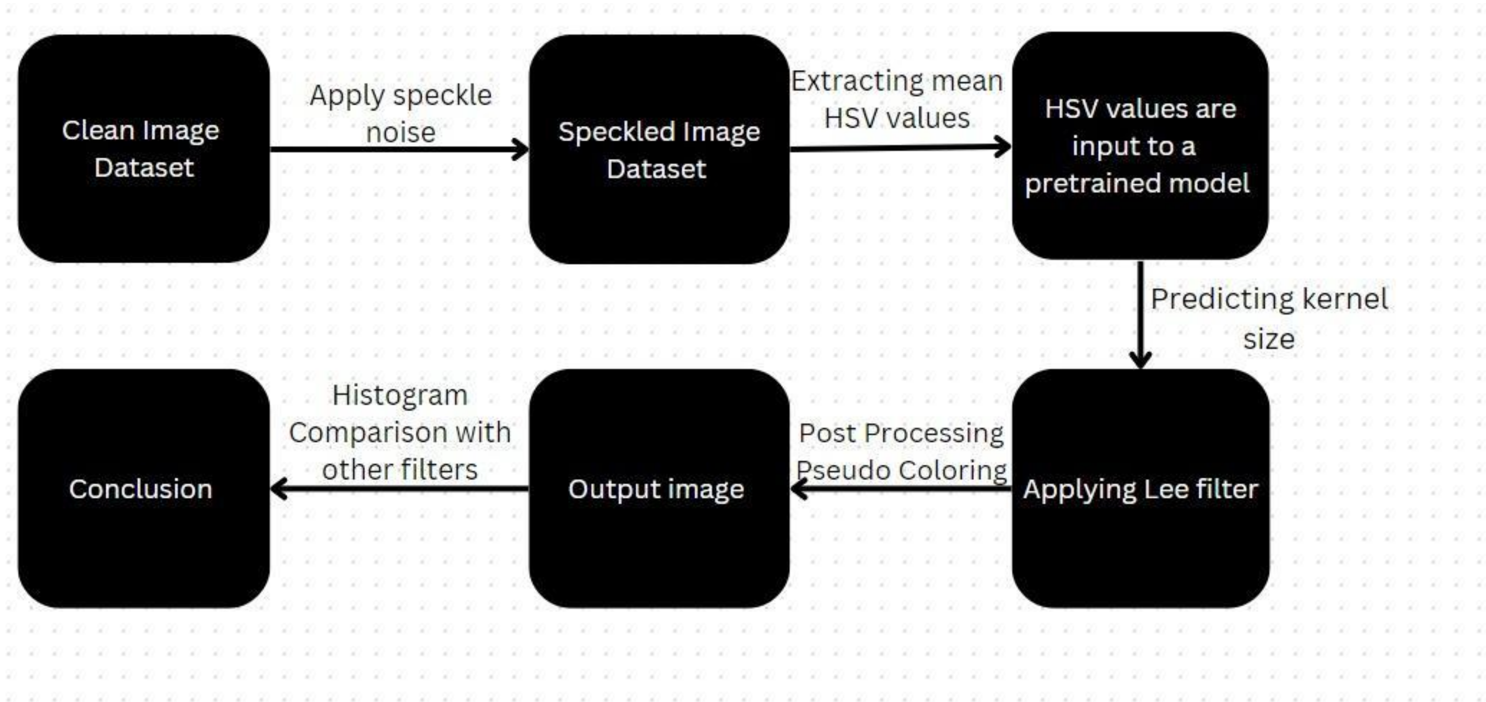
<u>Why does the kernel size matter ?</u>

- Kernel size picked for filtering is usually directly proportional to the noise density in an image. This means that larger kernel sizes would work better while filtering very noisy images.

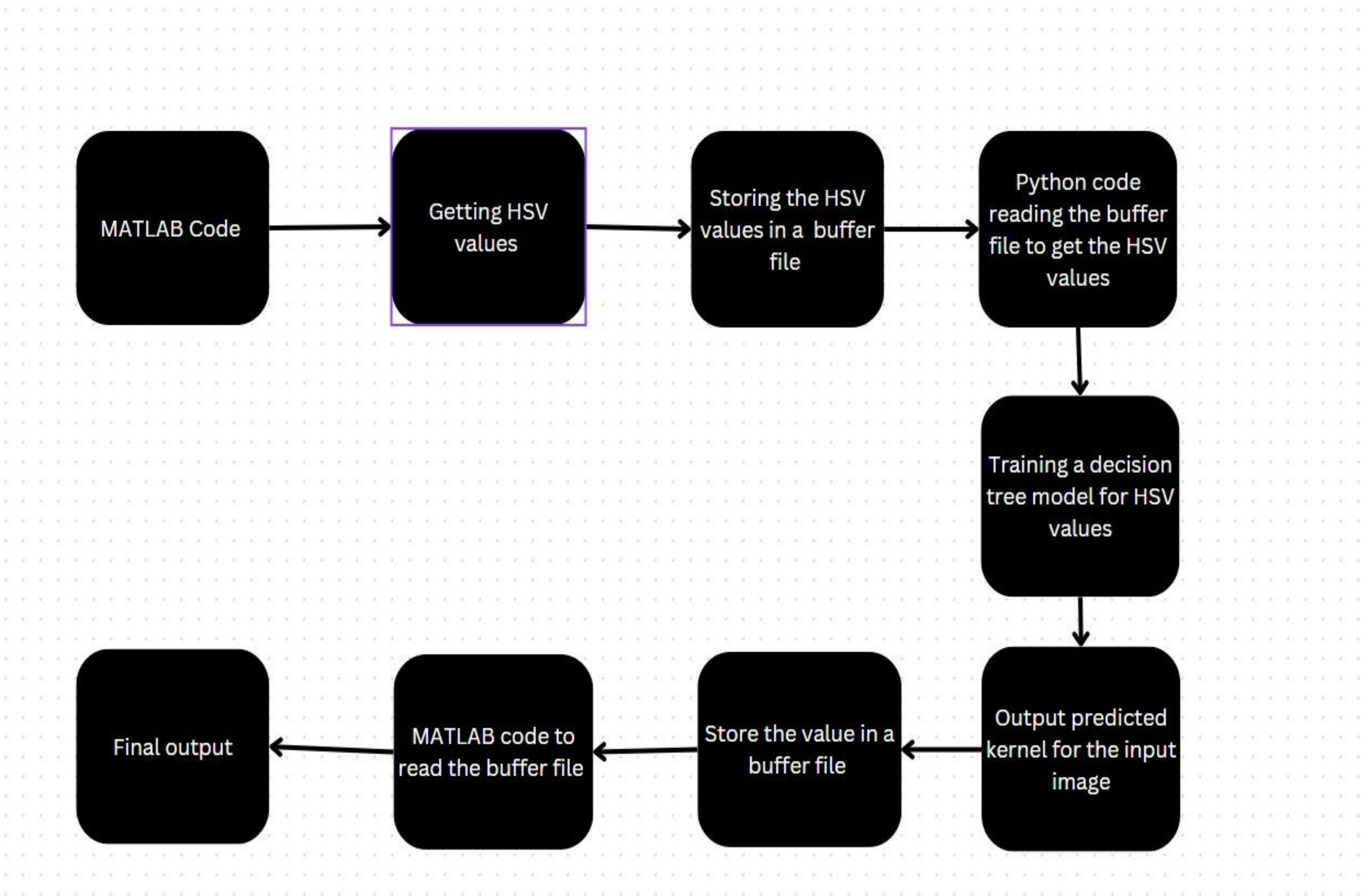
- Since noise in an image cannot be quantified without indirect methods, we analysed the HSV values in an image to find out the optimum kernel size.

- Initial data was created by tabulating the apt kernel size against the HSV values. This was later used to train an ML model.

Workflow:

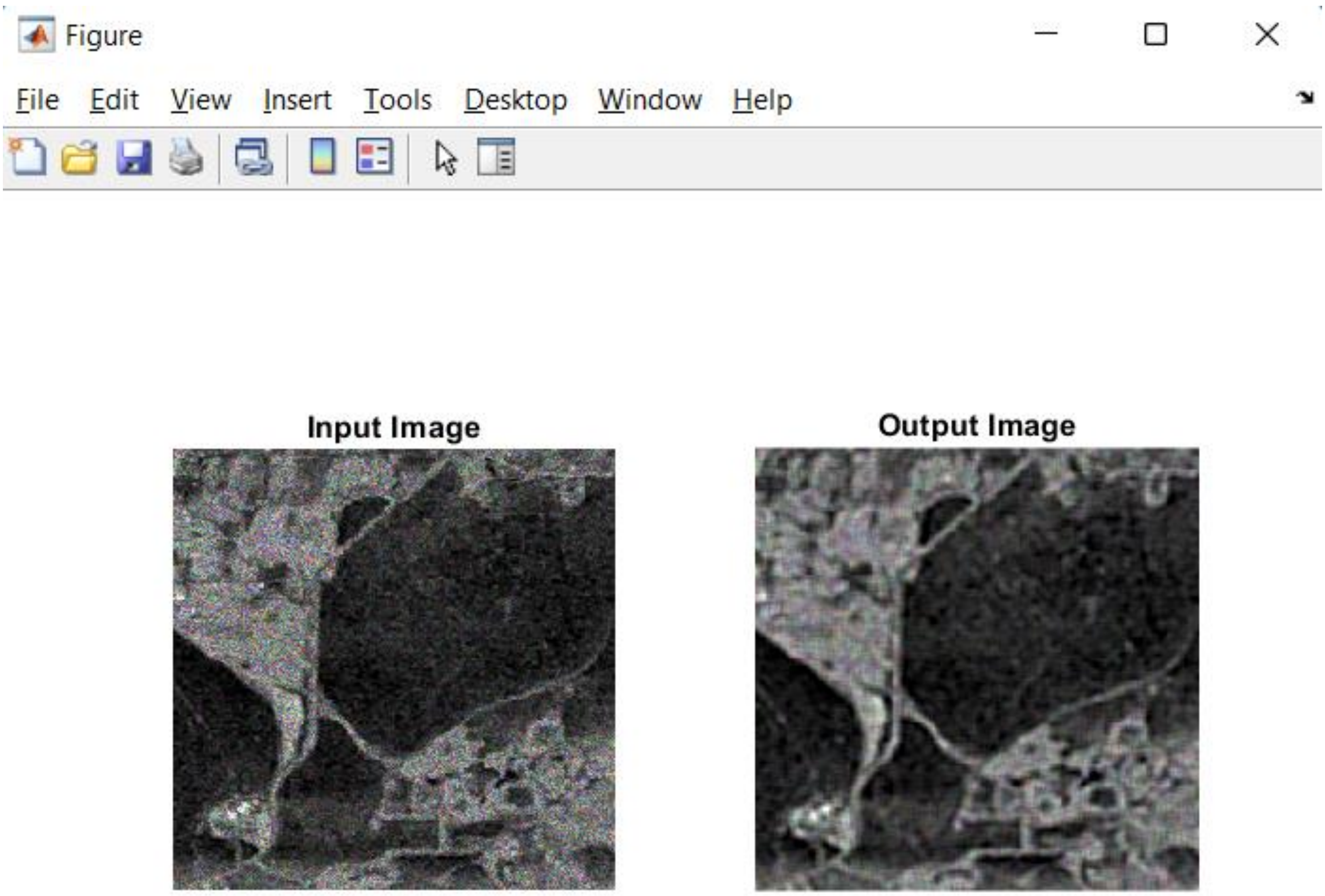


Implementation in Matlab and Python -



Results:

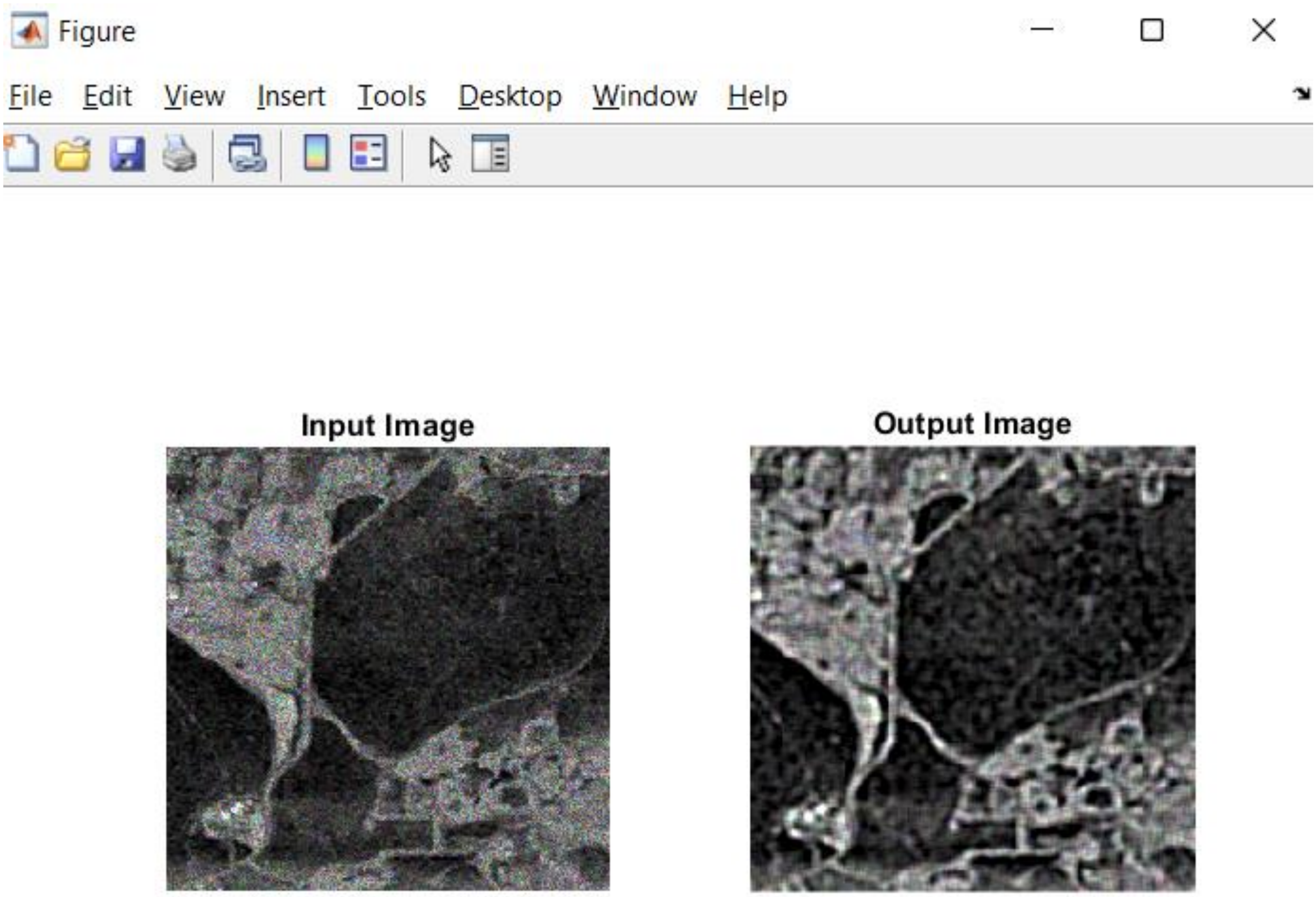
1) Lee Filtering result:



Observation of result 1:

- Speckle noise is removed.
- The image appears smoother, this can be seen especially on the lighter areas of the image.

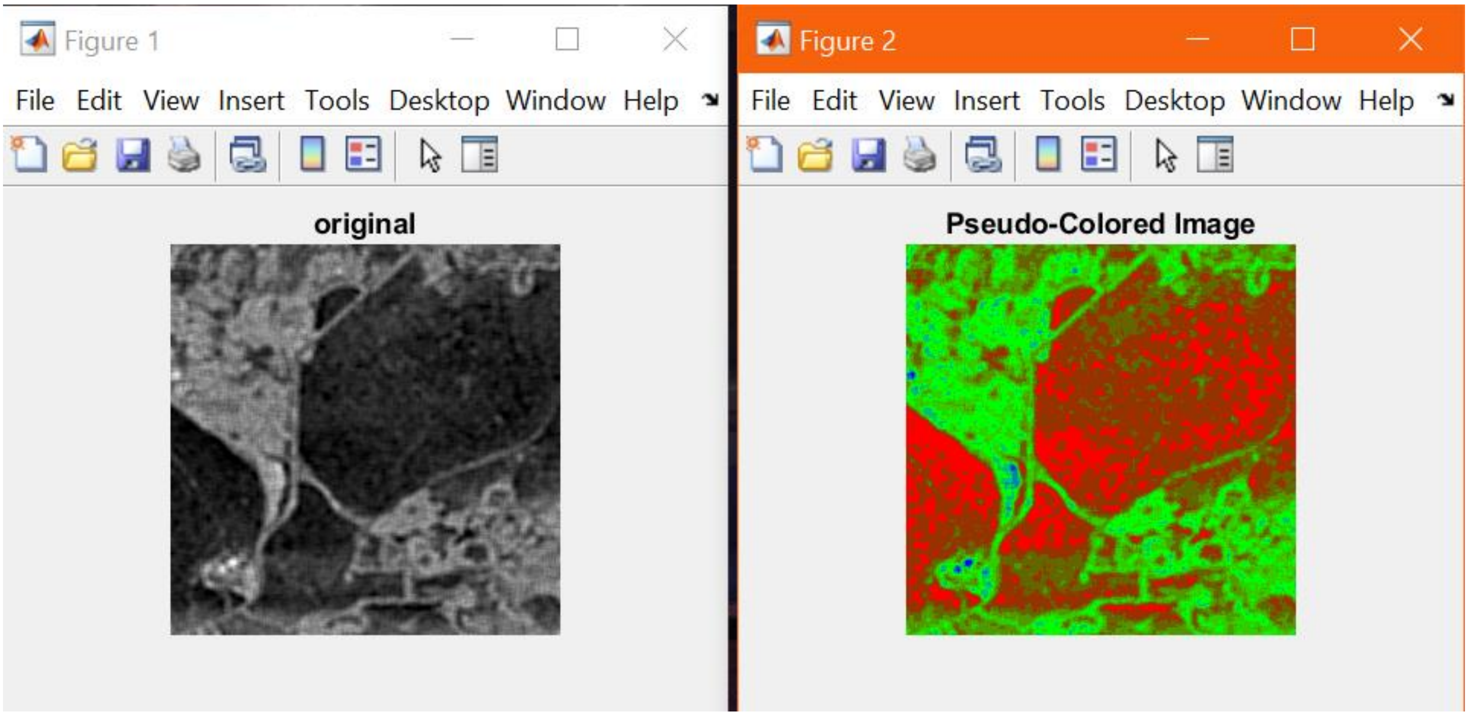
2) Post processing result:



Observation of result 2:

- Image is sharpened.
- The contrast in the image is increased, hence making it easier to distinguish borders and other details.

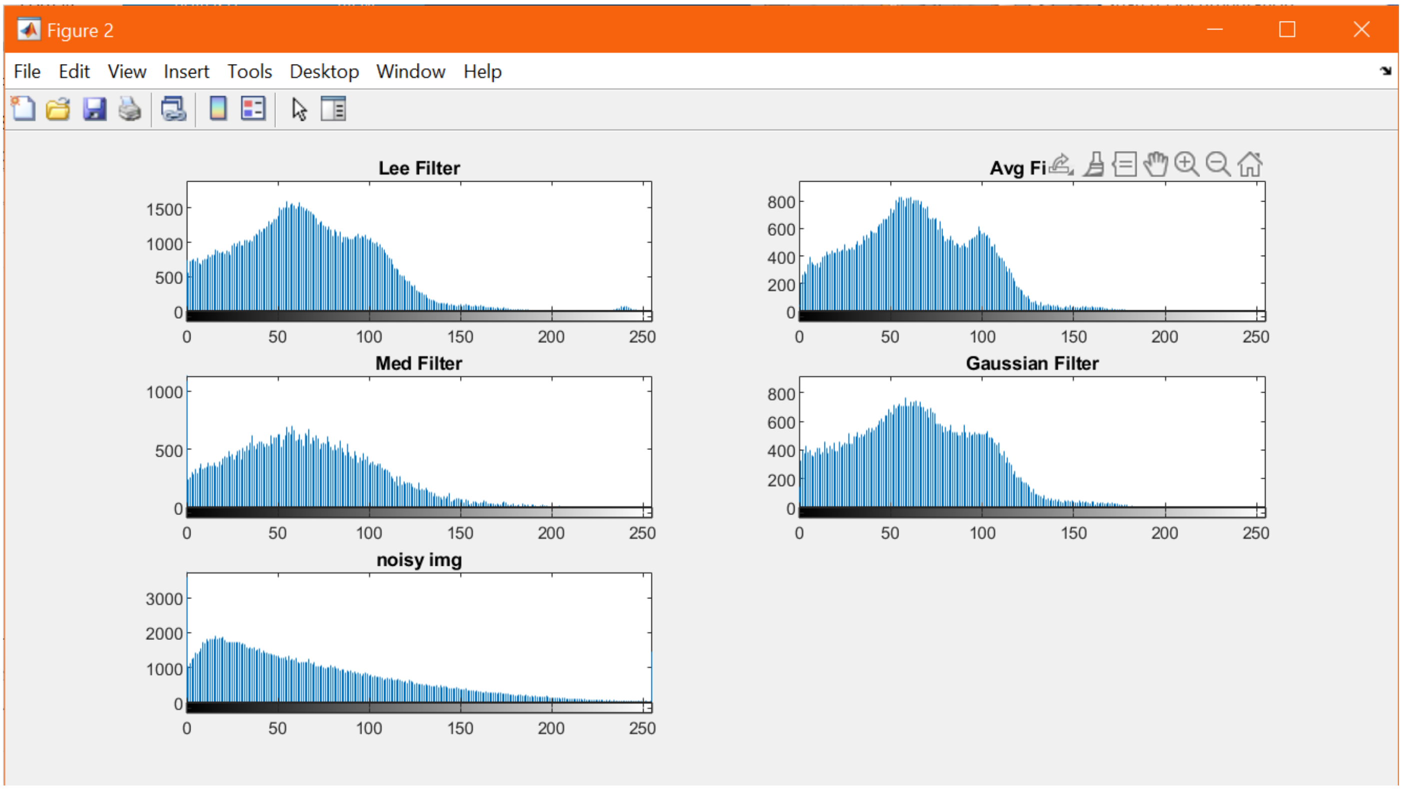
3) Pseudo colouring result:



Observation of result 3:

- Rougher terrain / terrain which is more absorbent to radio waves is darker and is thus more red in the pseudo coloring.
- Finer details in the image are easier to catch.

4) Histogram comparison :



Observation of result 4:

- Comparing the filter histograms with the noisy histogram we conclude that the Lee filter is better for SAR despeckling.
- The histograms of the output image and that of the original are very similar

Code :

Important sections of the code have been pasted below. For more code, the relevant files have been attached with this document.

Complete code - https://github.com/Rohanmrao/SAR-Image-Despeckling

Lee filter function -

```
function lee_output = Leefilter(img,window_size)


img = double(img);
lee_output = img;
means = imfilter(img, fspecial('average', window_size), 'replicate');
sigmas = sqrt((img-means).^2/window_size^2);
sigmas = imfilter(sigmas, fspecial('average', window_size), 'replicate');


ENLs = (means./sigmas).^2;
sx2s = ((ENLs.*(sigmas).^2) - means.^2)./(ENLs + 1);
fbar = means + (sx2s.*(img-means)./(sx2s + (means.^2 ./ENLs)));
lee_output(means~=0) = fbar(means~=0);


end
```

Image Sharpening –

```
function post_out = postpr(a)
    mf = ones(3, 3)/9;
    meanfilt = imfilter(a,mf);
    c =imsharpen(meanfilt,'Radius',3.5,'Amount',3.5);
    post_out = imfilter(c,mf);
end
```

Calculating HSV values of the image –

```
Storage =dir(fullfile("C:","Users","Rohan Mahesh
Rao","Desktop","DIP_project","Testset","speckled","*.png"));
fprintf("No.of images in the speckled set: %d\n",numel(Storage));


Noisy_set  = "C:\Users\Rohan Mahesh Rao\Desktop\DIP_project\Noisy_Dataset\";
Clean_set =  "C:\Users\Rohan Mahesh Rao\Desktop\DIP_project\Dataset\";
Testset_noisy = "C:\Users\Rohan Mahesh Rao\Desktop\DIP_project\Testset\speckled";
Testset_clean = "C:\Users\Rohan Mahesh Rao\Desktop\DIP_project\Testset\cleaned";
```

```
input = imread("C:\Users\Rohan Mahesh
Rao\Desktop\DIP_project\Testset\speckled\speckled3.png");
input_lee = im2double(input);


I_hsv = rgb2hsv(input);
hueval = 10*mean(mean(I_hsv(:,:,1)));
satval = 10*mean(mean(I_hsv(:,:,2)));
valval = 10*mean(mean(I_hsv(:,:,3))); % extracting hsv features of the image for it to act
as a unique image signature


hueval = round(hueval,1);
satval = round(satval,1);
valval = round(valval,1);


hueval = uint8(hueval);
satval = uint8(satval);
valval = uint8(valval);
formatSpec = '%d';
hsv_inputs = fopen("hsv_inputs.txt",'w');
fprintf(hsv_inputs,formatSpec,hueval,satval,valval); % writing hsv inputs to a buffer file
disp("HSV values written...");
```

Decision Tree Regression (Python) –

```
model1 = DecisionTreeRegressor(random_state = 1);
#training
model1.fit(x_train, y_train)
pred1 = model1.predict(x_test)


def get_kernel(x_given):
    return model1.predict(x_given)
hsv_inputs = open("hsv_inputs.txt","r")
x_inp = str(hsv_inputs.read())
x_given = []

for i in range (0,3):
    x_given.append(int(x_inp[i]))

print(x_given)

pred = get_kernel([x_given])
print("Predicted kernel size :",round(pred[0]))
mse_val = mse(y_test, pred1)
print("MSE: ",mse_val)
pred_file = open("predicted_kernel.txt","w")
pred_file.write(str(round(pred[0])))
pred_file.close() #to change file access modes
```

Pseudo Coloring –

```
function pseudo_image = Pseudo_Image(A)
    A = im2gray(A);
    [row,col]=size(A);
    for i=1:1:row
     for j=1:1:col
     if (A(i,j)>= 0) && (A(i,j) < 25)
     red(i,j)=255;
     green(i,j)=0;
     blue(i,j)=0;
     elseif (A(i,j)>= 25) && (A(i,j)< 50)
     red(i,j)=150;
     green(i,j)=51;
     blue(i,j)=0;
      elseif (A(i,j)>= 50) && (A(i,j)< 75)
     red(i,j)=102;
     green(i,j)=102;
     blue(i,j)=0;
      elseif (A(i,j)>= 75) && (A(i,j)< 100)
     red(i,j)=80;
     green(i,j)=153;
     blue(i,j)=0;
      elseif (A(i,j)>= 100) && (A(i,j)< 125)
     red(i,j)=51;
     green(i,j)=204;
     blue(i,j)=0;
      elseif (A(i,j)>= 125) && (A(i,j)< 150)
     red(i,j)=0;
     green(i,j)=255;
     blue(i,j)=0;
      elseif (A(i,j)>= 150) && (A(i,j)< 175)
     red(i,j)=0;
     green(i,j)=192;
     blue(i,j)=120;
      elseif (A(i,j)>= 175) && (A(i,j)< 200)
     red(i,j)=0;
     green(i,j)=129;
     blue(i,j)=180;
      elseif (A(i,j)>= 200) && (A(i,j)< 225)
     red(i,j)=0;
     green(i,j)=66;
     blue(i,j)=200;
     elseif (A(i,j) >= 225) && (A(i,j)< 255)
     red(i,j)=0;
     green(i,j)=0;
     blue(i,j)=255;
     end
     end
     end

    pseudo_image=cat(3,red,green,blue);
    pseudo_image=pseudo_image/255;%convert from 0-255 to 0-1

end
```

References :

https://ieeexplore.ieee.org/document/9399231

https://crisp.nus.edu.sg/~research/tutorial/sar_int.htm

https://www.kaggle.com/code/samvram/flood-detection-sar/data