BIRLA INSTITUTE OF TECHONOLOGY
DEPARTMENT OF EEE

DIGITAL SIGNAL PROCESSING PROJECT-
COLOURED IMAGE DENOISING FILTER
(USING MATLAB) BY:-

1) PRE-SWITCHING MEDIAN FILTER
2) FOURIER TRANSFORM FILTER
3) K-L TRANSFORM BASED FILTER

UNDER THE GUIDANCE OF –

DR. GAURI SHANKER GUPTA

Aryan Sinha
By – ARYAN SINHA (BTECH/10227/21)

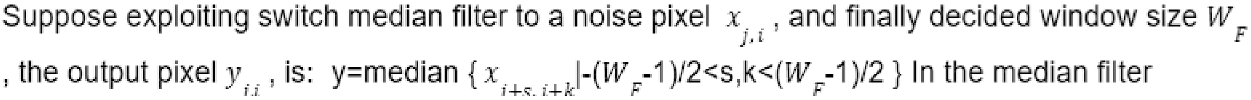
BRIEF THEORY AND EXPLANATION OF ALGORITHM FOR IMPLEMENTATION PRE-SWITCHING MEDIAN FILTER FOR COLOURED IMAGE DENOISING.

THEORY AND METHODOLOGY:

Pre-switching median filter is an image denoising filter that uses a combination of median filter and mean filter. The filter operates on small neighborhoods of pixels in the image, and replaces the central pixel with the median value of the pixels in the neighborhood if the difference between the median value and the original pixel value exceeds a certain threshold. Otherwise, the central pixel is replaced with the mean value of the neighborhood. The idea behind the pre-switching median filter is to combine the advantages of median and mean filters. Median filters are good at removing impulse noise, but they tend to produce blurring effects, especially when the noise is not isolated. On the other hand, mean filters are good at removing Gaussian noise, but they tend to preserve the edges of the image poorly. The pre-switching median filter overcomes these limitations by switching between the median and mean filters based on the level of noise in the neighborhood.

LITERATURE REVIEW:

Pre-switching median filter is an image-denoising filter that uses a combination of median filter and mean filter. The proposed noise removal method lowers the number of pixels that are submitted to median filtration. The approach includes a switching mechanism that decides the action to be performed on consecutive pixels of the filtered image. Only faulty pixels are subjected to median filtration. Uncorrupted pixels are not altered. [3]



Suppose exploiting switch median filter to a noise pixel $x_{j,i}$, and finally decided window size W_F , the output pixel $y_{i,i}$, is: y=median { $x_{i+s,i+k}$|-(W_F-1)/2<s,k<(W_F-1)/2 } In the median filter

In the median filter process, those corrupted pixels are included; only those corrupted ones are considered to get the median value. If the current pixel being processed is less than the threshold value, it is ignored. Otherwise, a new pixel value computed using the suggested algorithm will take its place. [5] The proposed filter operates in two stages. In the first stage, the image is divided into non-overlapping blocks, and the median value of each block is determined. In the second stage, the blocks are sorted based on the magnitude of their median values, and a threshold is applied to select only the blocks with median values above a certain threshold. The selected blocks are then filtered using a standard median filter. [1]

ALGORITHM:

1) Read the input image and reference image and define a threshold value for pixel comparison.
2) Define the size of the filter window.
3) Initialize a variable to keep track of the number of pixels modified during filtering.
4) Initialize the output filtered image as the input noisy image.
5) Traverse through each pixel of the image and for each color channel:

   a. Extract the filter window around the current pixel.

   b. Sort the values in the filter window.

   c. Calculate the median value of the filter window.

   d. Compare the median value with the original pixel value.

   e. If the difference between the median value and the original pixel value is greater than the threshold, replace the original pixel value with the median value and increment the modified pixel count.

6) Calculate the performance metrics (MSE, RMSE, PSNR, SSIM) between the reference and filtered images.
7) Display the performance metrics and the original and filtered images.

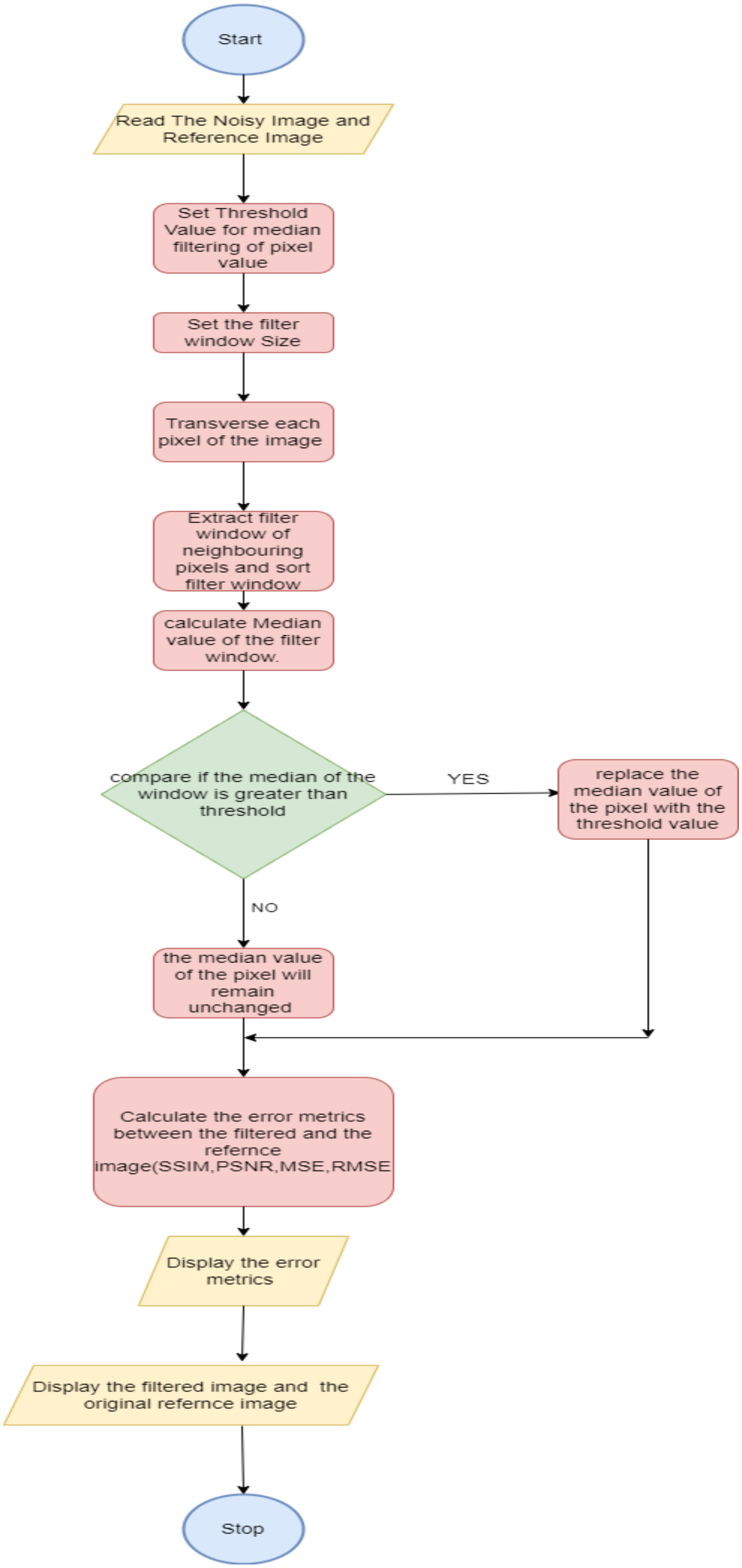
PSEUDO CODE:

```
% Read image and define switching threshold
I = imread('IMAGE.png');
ref_image = imread('REF_IMG.jpg');
ref_image = imresize(ref_image, [size(I,1) size(I,2)]);
threshold = 80;

% Define filter window size
filter_size = 3;

ier=0;

% Initialize output image
psmf_filtered_image = I;

% Traverse through each pixel of the image
[rows, cols, channels] = size(I);
for c = 1:channels
   for i = 1:rows
      for j = 1:cols
         % Extract filter window
         filter_window = zeros(filter_size*filter_size, 1);
         idx = 1;
         for k = max(1, i-filter_size):min(i+filter_size, rows)
```

```
                for l = max(1, j-filter_size):min(j+filter_size, cols)
                    filter_window(idx) = I(k, l, c);
                    idx = idx + 1;
                end
            end
            filter_window = filter_window(1:idx-1);
            % Sort filter window
            sorted_window = sort(filter_window);
            % Calculate median value of filter window
            if mod(numel(sorted_window), 2) == 0
                median_val = mean(sorted_window(numel(sorted_window)/2:numel(sorted_window)/2+1));
            else
                median_val = sorted_window(ceil(numel(sorted_window)/2));
            end
            % Compare to original pixel value
            if abs(median_val - I(i, j, c)) > threshold
                ier = ier + 1;
                psmf_filtered_image(i, j, c) = median_val;
            end
        end
    end
end

% Calculate performance metrics between reference and filtered images
mse = sum(sum(sum((double(ref_image) - double(psmf_filtered_image)).^2)))/(rows*cols*channels);
rmse = sqrt(mse);
max_i = double(max(ref_image(:)));
psnr = 20*log10(max_i/rmse);
[ssim_index, ~] = ssim(ref_image, psmf_filtered_image);

% Display performance metrics
fprintf('MSE: %.2f, RMSE: %.2f, PSNR: %.2f dB, SSIM: %.4f, IER: %d\n', mse, rmse, psnr, ssim_index,
ier);
% Display original and filtered images
figure;
subplot(1,3,1);imshow(I);title('Noisy Image');
subplot(1,3,2);imshow(ref_image);title('Reference Image');
subplot(1,3,3);imshow(psmf_filtered_image);title('Filtered Image');
```

FLOWCHART DESCRIPTION:

The given code implements a Pixel-Sorted Median Filter (PSMF) for image denoising. The code first reads in an image and a reference image, and defines a switching threshold value. A filter window size is also defined. The code then initializes an output image, and traverses through each pixel of the input image. For each pixel, a filter window is extracted, and the median value of the filter window is calculated. The median value is compared to the original pixel value, and if the difference between the two values exceeds the switching threshold, the median value is used as the filtered pixel value. The code then calculates several performance metrics between the reference and filtered images, including mean squared error (MSE), root mean squared error (RMSE), peak signal-to-noise ratio (PSNR), and structural similarity index(SSIM).

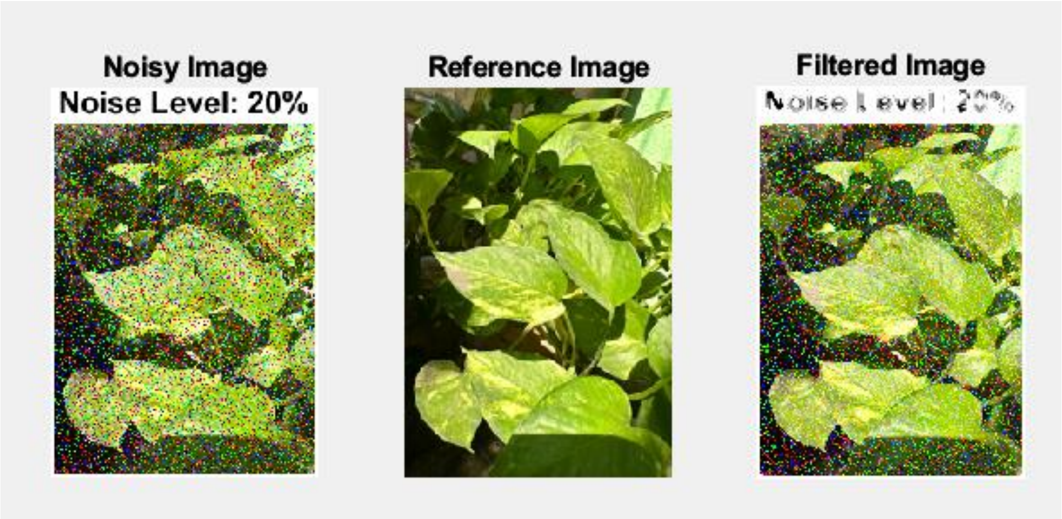

Start
Read The Noisy Image and Reference Image
Set Threshold Value for median filtering of pixel value
Set the filter window Size
Transverse each pixel of the image
Extract filter window of neighbouring pixels and sort filter window
calculate Median value of the filter window.
compare if the median of the window is greater than threshold
YES
replace the median value of the pixel with the threshold value
NO
the median value of the pixel will remain unchanged
Calculate the error metrics between the filtered and the refernce image(SSIM,PSNR,MSE,RMSE
Display the error metrics
Display the filtered image and the original refernce image
Stop

Advantages and Disadvantages of Pre-Switching Median filter over a Median Filter

Better edge preservation: The pre-switching median filter can preserve edges better than the traditional median filter, thanks to the pre-switching mechanism that can differentiate between noise and edges. Higher denoising efficiency: The pre-switching median filter is more efficient in removing noise from images than the traditional median filter, especially in images with high levels of noise. Reduced halo effect: The pre-switching median filter reduces the halo effect that can occur around edges in the traditional median filter. Better detail preservation: The pre-switching median filter can better preserve the details in images than the traditional median filter, especially in the presence of high-frequency noise. Reduced computational complexity: The pre-switching median filter reduces the computational complexity of the filtering process compared to other advanced image denoising algorithms, making it a more efficient option for real-time applications.

Increased memory requirements: The pre-switching median filter requires more memory than the traditional median filter due to the additional calculations and storage needed for the pre-switching mechanism. Reduced processing speed: The pre-switching median filter requires more processing time than the traditional median filter due to the additional calculations involved in the pre-switching mechanism, which can lead to slower processing speeds.

Sample output :(for 20% impulse Noise)



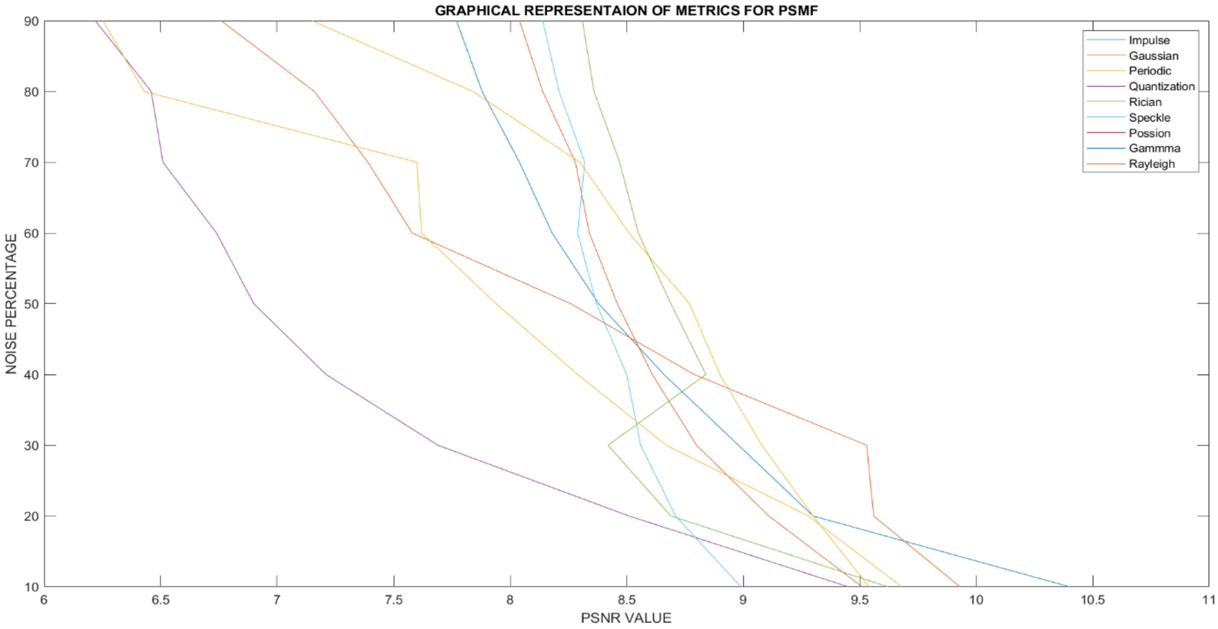
Error metrics: MSE :10194.26, RMSE: 100.97, PSNR: 8.05 dB, SSIM: 0.3016

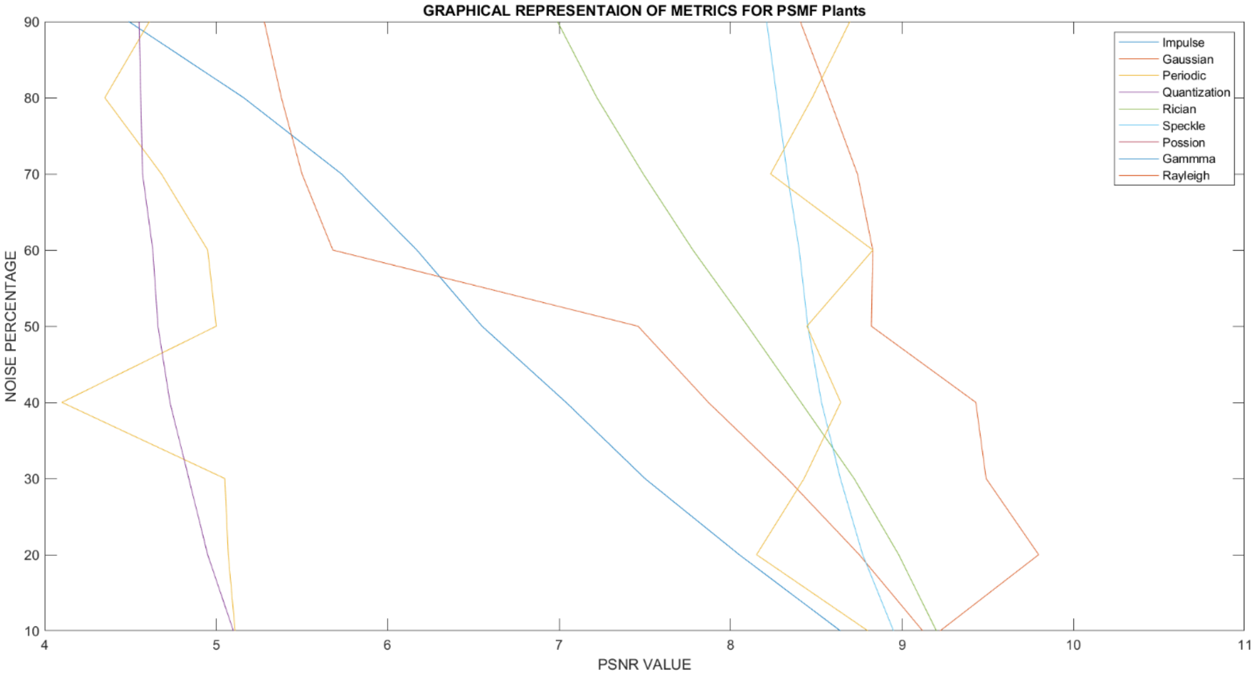
Variable and Parameter Description

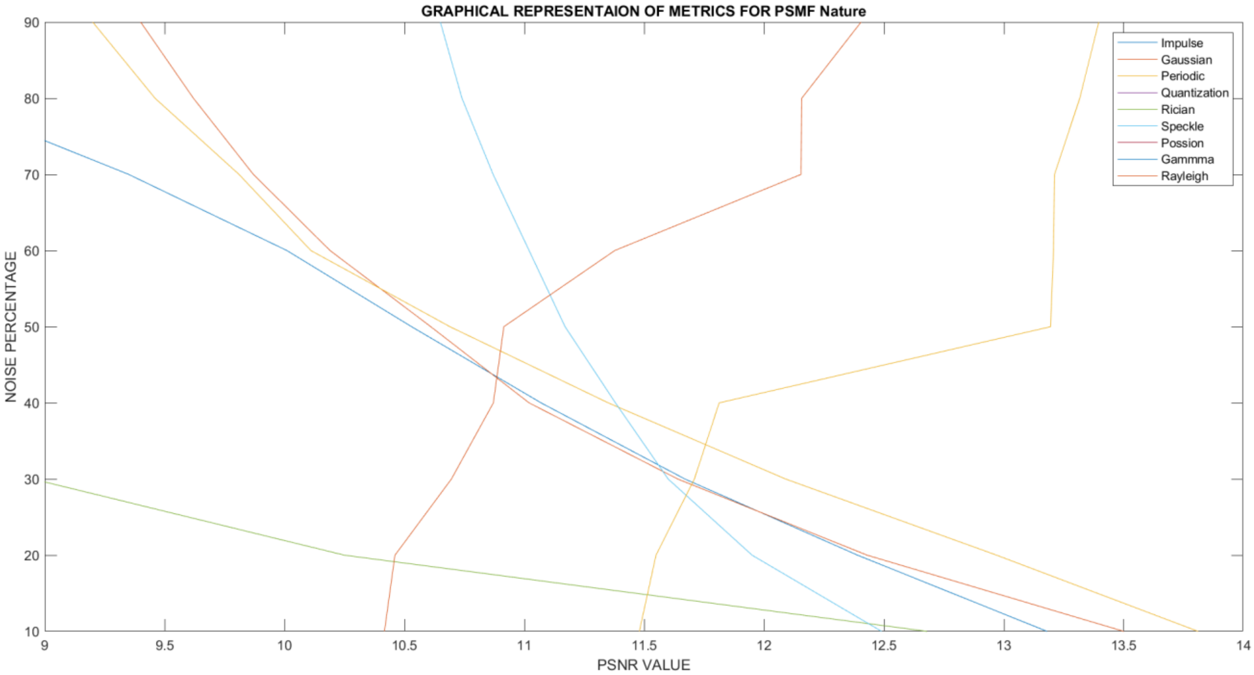
Variable			**Description**
I			Input image to be filtered
ref_image			Reference image for calculating performance metrics
threshold			Switching threshold for pre-switching mechanism.
filter_size			Size of the filter window.
psmf_filtered_image			Output image after filtering with pre-switching median filter.
rows			Number of rows in the input image.
cols			Number of columns in the input image.
channels			Number of color channels in the input image.
c			Index for the color channel being processed in the loop.
i			Index for the row being processed in the loop.
j			Index for the column being processed in the loop.
filter_window			Array containing the pixels in the filter window.
idx			Index variable for the filter_window array
k			Index variable for the rows in the filter window.

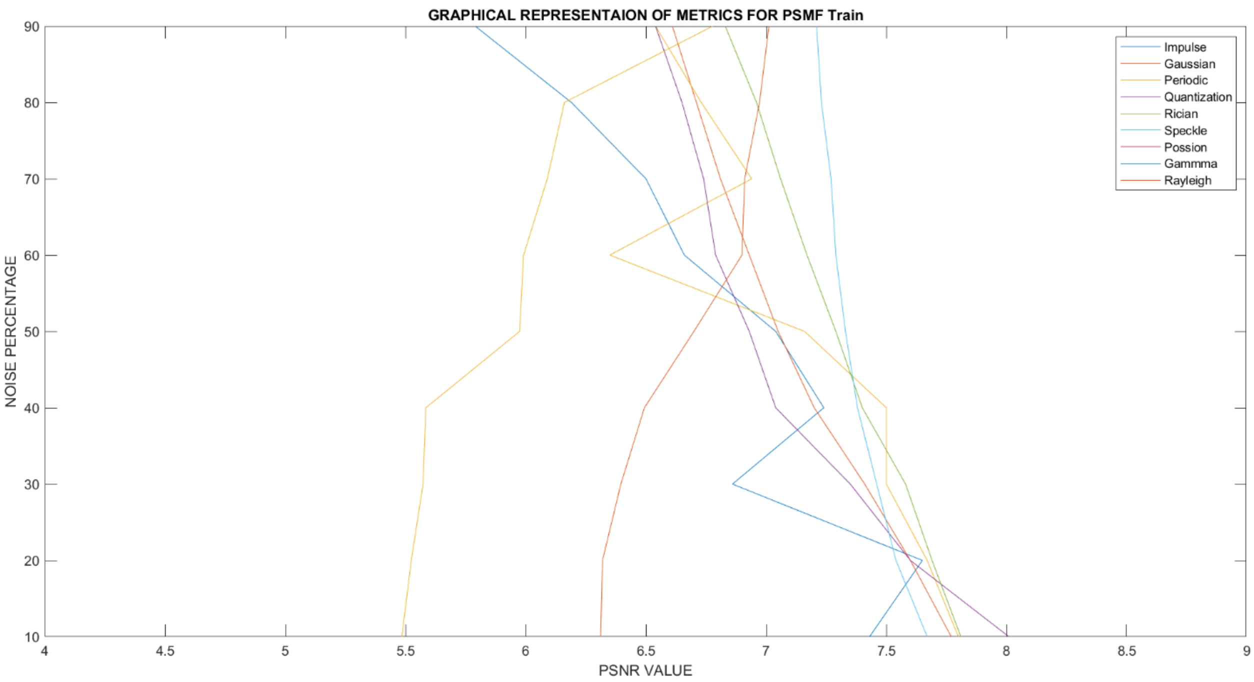
l			Index variable for the columns in the filter window
sorted_window			Sorted array of pixel values in the filter window
median_val			Median value of the pixel values in the filter window.
mse			Mean squared error between reference and filtered images.
rmse			Root mean squared error between reference and filtered images.
max_i			Maximum pixel value in the reference image.
psnr			Peak signal-to-noise ratio between reference and filtered images.
ssim_index			Structural similarity index between reference and filtered images.

METRICS GRAPHICAL REPRESENTATION (NOISE PERCENT VS PSNR)




GRAPHICAL REPRESENTAION OF METRICS FOR PSMF Plants
NOISE PERCENTAGE
90
80
70
60
50
40
30
20
10
4
5
6
7
8
9
10
11
PSNR VALUE
Impulse
Gaussian
Periodic
Quantization
Rician
Speckle
Possion
Gammma
Rayleigh


GRAPHICAL REPRESENTAION OF METRICS FOR PSMF Nature
NOISE PERCENTAGE
90
80
70
60
50
40
30
20
10
9
9.5
10
10.5
11
11.5
12
12.5
13
13.5
14
PSNR VALUE
Impulse
Gaussian
Periodic
Quantization
Rician
Speckle
Possion
Gammma
Rayleigh



CONCLUSION (BASED ON ABOVE GRAPHICAL RESULTS)

The proposed filter operates in two stages. The first stage selects only the blocks with median values above a certain threshold, and the second stage applies a standard median filter to the selected blocks The median filter is commonly used for image denoising because **it is effective at removing impulse noise**, which is characterized by large, isolated spikes in the image**.** Therefore, the proposed filter can be useful for image-denoising applications. The filter is also works gaussian to some extent. From the above results of the graph, The filter seems to be working the least on quantization and periodic noise. For other noises the filter works average. We can also see that in this filtering technique used spikes are seen of sudden change of PSNR can be seen on different noises for different images. This mostly depends on the pixel value of the image (i.e Either the image is taken in low light or bright light).

RESEARCH PAPERS REFRENCES:

1) *Zhou Wang, & Zhang, D. (1999). Pre-switching median filter for the removal of impulse noise from highly corrupted images. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 46(1), 78–80. doi:10.1109/82.749102 LINK:HERE*

2) *Modified switching median filter with one more noise detector for impulse noise removal Chung-Chia Kanga, Wen-June Wanga,b,∗ Department of Electrical Engineering, National Central University, Jhongli, Taoyuan 32001, Taiwan, ROC department of*

Electrical Engineering, National Taipei University of Technology, Taipei 10608, Taiwan, ROC Received 29 May 2008; accepted 5 August 2008. LINK:HERE

3) *Noise adaptive switching median-based filter for impulse noise removal from extremely corrupted images A. Fabijan´ ska D. Sankowski Department of Computer Engineering, Technical University of Lodz, 18/22 Stefanowskiego Street, 90-924 Lodz, Poland. LINK:HERE*

4) *A New Switching-Based Median Filtering Scheme and Algorithm for Removal of High-Density Salt and Pepper Noise in Images V. Jayaraj and D. Ebenezer Digital Signal Processing Laboratory, Sri Krishna College of Engineering and Technology, Coimbatore, Anna University Coimbatore, Tamilnadu 641008, India LINK:HERE*

5) *Fast and reliable switching median filter for highly corrupted images by impulse noise Wei Ping* , Li Junli† , Lu Dongming* , Chen Gang+ * Department of Computer Science and Engineering, Zhejiang University, Hangzhou, China LINK:HERE*

BRIEF THEORY AND ALGORITHM OF CODE FOR IMPLEMENTATION FOURIER TRANSFORM-BASED COLOURED IMAGE DENOISING FILTER.

THEORY AND METHODOLOGY:

In the context of signals and systems, the Fourier transform is used to represent a signal as a sum of sine and cosine waves of different frequencies. This allows us to analyze the signal in terms of its frequency content, and is particularly useful for understanding the behavior of filters and other linear systems. Using the FFT, the Fourier Transform-based filter transforms the input image from the spatial domain to the frequency domain. An image's frequency domain representation represents the image's various frequency components, where the lower frequencies correspond to the smooth regions of the image and higher frequencies correspond to the edges and details of the image. By filtering out the high-frequency components using a low-pass filter, the filter effectively removes the noise from the image. After filtering, the image is transformed back to the spatial domain using the IFFT, resulting in a denoised image.

LITERATURE REVIEW:

Fourier transform is a mathematical technique that transforms a signal from the time domain to the frequency domain. It is used in many fields, including mathematics, physics, engineering, and signal processing. The Fourier transform is named after the French mathematician Joseph Fourier, who first introduced the concept in the early 19th century. Low pass filtering is used in spatial-frequency domain filtering techniques, which involves creating a frequency domain filter that passes all frequencies below a cut-off frequency and attenuates all frequencies above it. Image information typically spreads in the low-frequency domain after being converted by low-pass filters like the Fourier transform, whereas noise typically spreads in the high-frequency domain. So, by choosing particular transform domain features and translating them back to the image domain, we may eliminate noise [2]. Here and (-1) x+y, j is the shifting and imaginary components, respectively. The filter restores the identified noisy frequencies during the algorithm's filtering phase. The repaired image is finally recreated using inverse shifting and inverse Fourier transform techniques. If I is the origin-shifted Fourier processed frequency domain image of the input corrupted image of size M X N, F is defined as

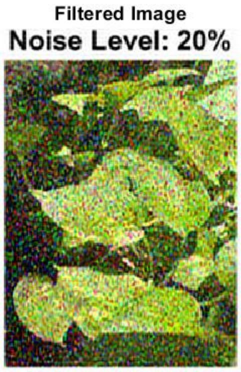
$$F(u,v)= 1/MN \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (-\ 1)^{x+y} I(x,y) e^{-j2\pi((ux/M)+(vy/N))}$$

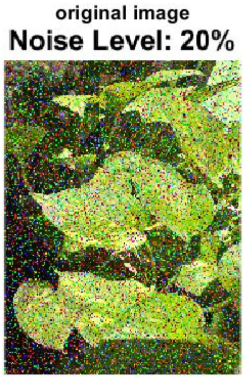
[1].[1]The discrete Fourier transform (DFT) is the discrete version of the continuous Fourier transform, which is used to transform a finite sequence of equally spaced samples of a signal from the time domain to the frequency domain. The DFT is defined as DFT: F(k) =$\sum_{n=0}^{n=N-1}$ f(n) $e^{-j2\pi kn/N}$where f(n) is the nth sample of the discrete signal, F(k) is the kth sample of the DFT of the signal, N is the total number of samples, and j is the imaginary unit.The inverse Fourier transform is used to transform a signal from the frequency domain back to the time domain. The inverse Fourier transform of F(w) is given by:DFT Inverse: f(n) = 1/N $\sum_{k=0}^{k=N-1}$ F(k) $e^{j2\pi kn/N}$where f(t) and f(n) are the original signals in the time domain, F(w) and F(k) are the signals in the frequency domain, and j is the imaginary unit.**[5]**

ALGORITHM:

1) Read the input image.
2) Convert the input image to double.
3) Display the original image.
4) Apply Fourier Transform to each color channel of the input image.
5) Define a low-pass filter using the circular equation.
6) Resize the filter to match the dimensions of the Fourier Transform.
7) Apply the filter to the frequency domain of each color channel.
8) Inverse shift the filtered color channels.
9) Combine the filtered color channels to create the filtered image.
10) Display the filtered image.
11) Calculate the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM) between the noisy and denoised images.
12) Display the performance metrics.

Sample output : (For 20% Impulse Noise)





Error Metrics: MSE: 0.04, RMSE: 0.20, PSNR: 13.80 dB, SSIM: 0.6872

Variable and Parameter Description

VARIABLE			**DESCRIPTION**
I			the original image loaded from file and converted to double precision
R, G, B			Fourier Transform of the all 3 channels of the original image
x,y			dimensions of the first two dimensions of the original image
r			radius of the low-pass filter
low_pass_filter			a matrix representing a circular low-pass filter in the frequency domain
Rshift, Gshift, Bshift			R ,G, B with the zero-frequency component shifted to the center of the array
Rfiltered_shift, Gfiltered_shift, Bfiltered_shift			R,G,B shift filtered by the low-pass filter in the frequency domain

Rfiltered, Gfiltered ,Bfiltered			R,B, G filtered_shift with the zero-frequency component shifted back to the top-left corner of the array
filtered_image			the denoised image constructed by taking the inverse Fourier Transform of Rfiltered, Gfiltered, and Bfiltered
mse			mean-squared error between the original and filtered images
rmse			root-mean-squared error between the original and filtered images
max_i			maximum pixel value of the original image
psnr			peak signal-to-noise ratio in decibels between the original and filtered images
ssim_index			structural similarity index between the original and filtered images

PSEUDO CODE

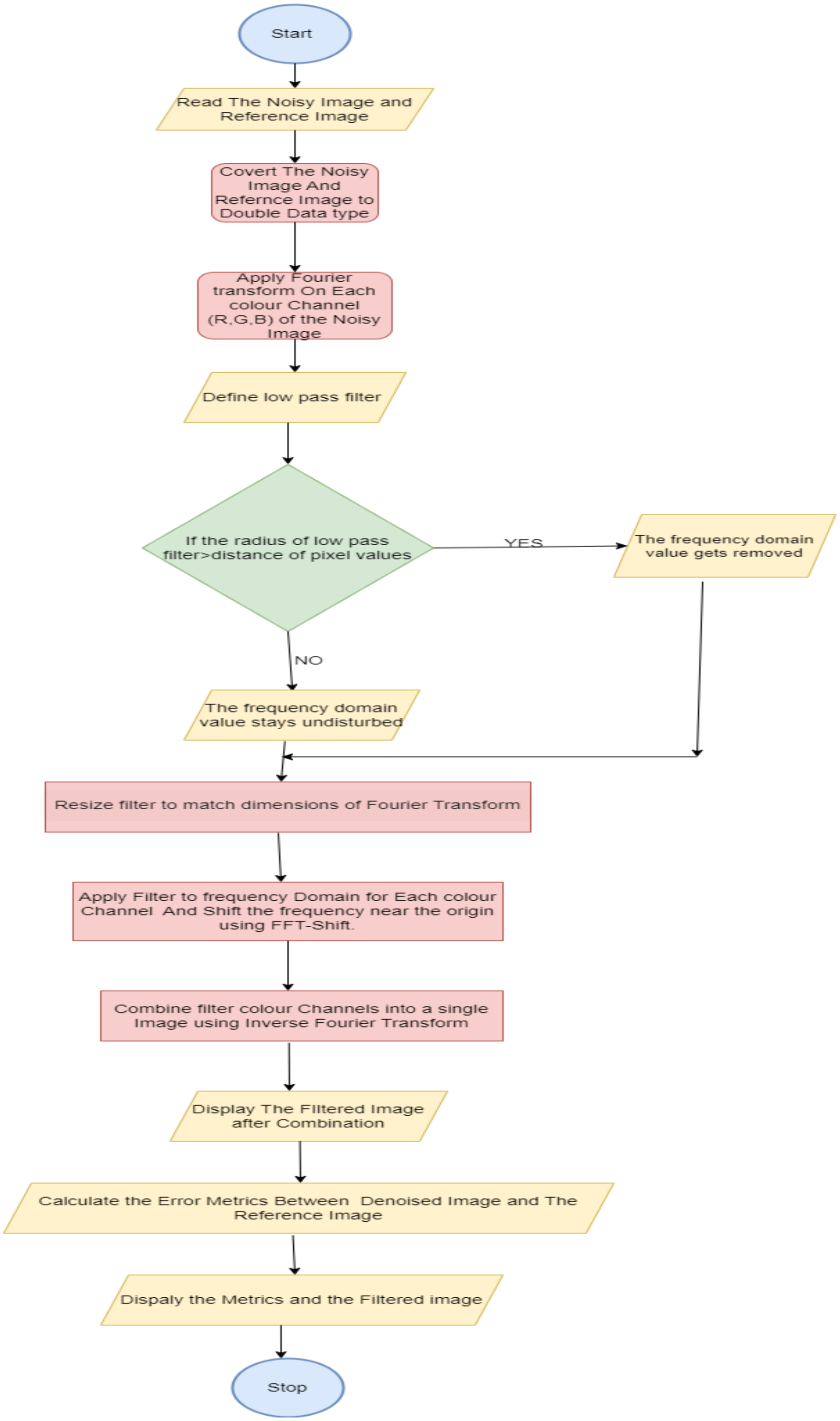
```
% Read image
I = imread('impulse_20-plants.png');
I = im2double(I);

% Display original image
figure;imshow(I);title("original image");

% Apply Fourier Transform to each color channel
R = fft2(I(:,:,1));
G = fft2(I(:,:,2));
B = fft2(I(:,:,3));

% Define a low-pass filter
[x, y] = size(I(:,:,1));
r = 75;
low_pass_filter = zeros(x, y);
```

```
for i = 1:x
    for j = 1:y
        if ((i-x/2)^2 + (j-y/2)^2)^(1/2) < r
            low_pass_filter(i, j) = 1;
        end
    end
end

% Resize filter to match dimensions of Fourier Transform
low_pass_filter = imresize(low_pass_filter, [size(I(:,:,1), 1), size(I(:,:,1), 2)]);

% Apply filter to frequency domain for each color channel
Rshift = fftshift(R);
Gshift = fftshift(G);
Bshift = fftshift(B);
Rfiltered_shift = Rshift .* low_pass_filter;
Gfiltered_shift = Gshift .* low_pass_filter;
Bfiltered_shift = Bshift .* low_pass_filter;
Rfiltered = ifftshift(Rfiltered_shift);
Gfiltered = ifftshift(Gfiltered_shift);
Bfiltered = ifftshift(Bfiltered_shift);
filtered_image = cat(3, real(ifft2(Rfiltered)), real(ifft2(Gfiltered)), real(ifft2(Bfiltered)));

% Display filtered image
figure;imshow(filtered_image);title('Filtered Image');

% Calculate performance metrics between noisy and denoised images
mse = sum(sum(sum((double(I) - filtered_image).^2)))/(x*y*3);
rmse = sqrt(mse);
max_i = double(max(I(:)));
psnr = 20*log10(max_i/rmse);
ssim_index = ssim(I, filtered_image);

% Display performance metrics
fprintf('MSE: %.2f, RMSE: %.2f, PSNR: %.2f dB, SSIM: %.4f', mse, rmse, psnr, ssim_index);
```

FLOWCHART DESCRIPTION

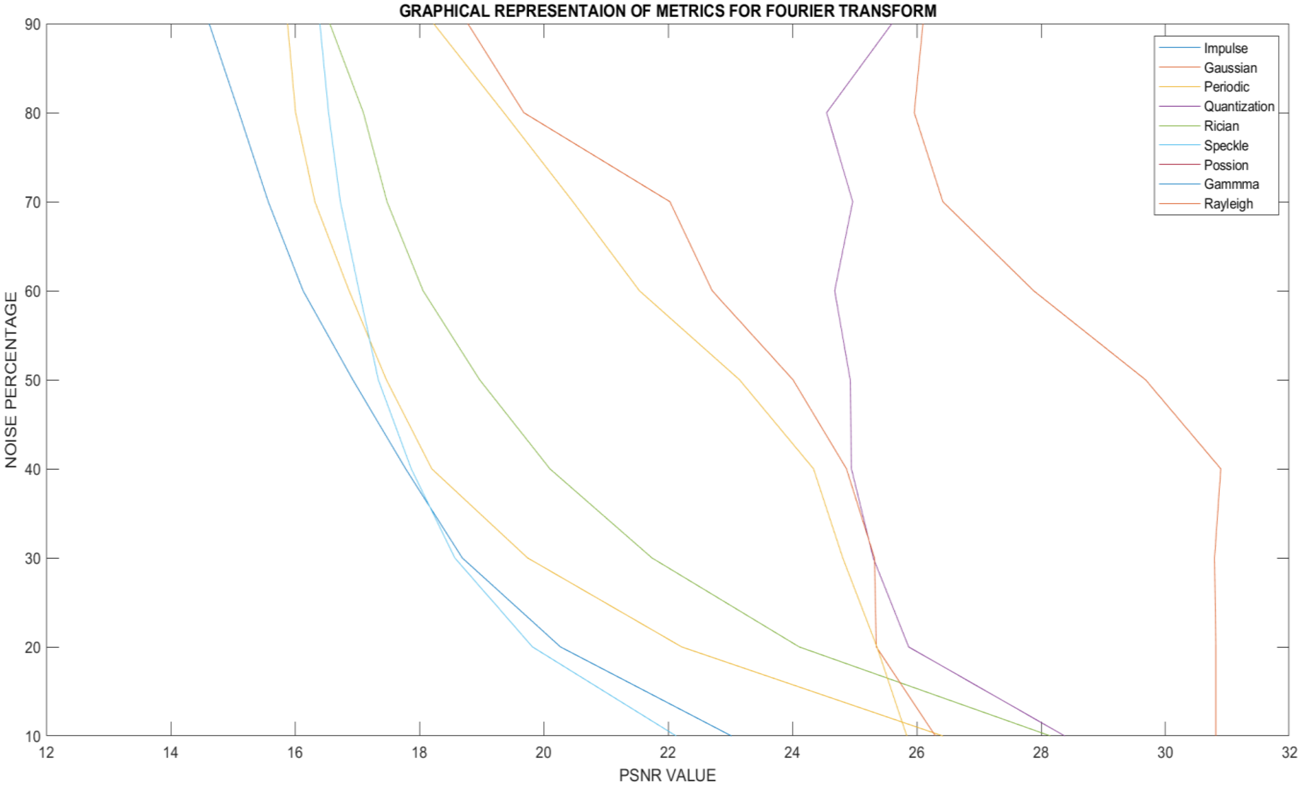
The above code performs image filtering in the frequency domain using a low-pass filter to remove high-frequency noise from an input image. The input image is first read and converted to double format. Then, the Fourier Transform is applied to each color channel separately. A low-pass filter is defined using a circular mask, and the mask is resized to match the dimensions of the Fourier Transform. The filter is then applied to the frequency domain for each color channel using element-wise multiplication with the shifted Fourier Transform. Finally, the filtered image is obtained by inverse Fourier Transform of each color channel and the performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM) are calculated to evaluate the filtering performance. The filtered image and performance metrics are displayed at the end of the code.

FLOWCHART

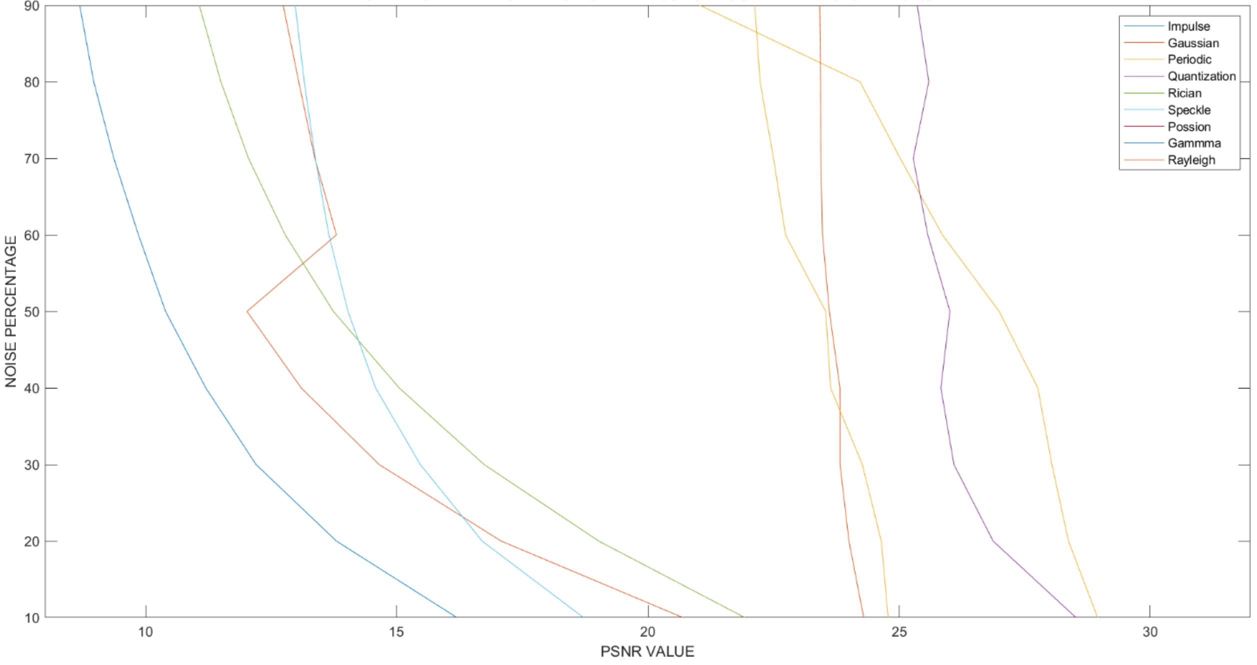

Start
Read The Noisy Image and Reference Image
Covert The Noisy Image And Refernce Image to Double Data type
Apply Fourier transform On Each colour Channel (R,G,B) of the Noisy Image
Define low pass filter
If the radius of low pass filter>distance of pixel values
YES
The frequency domain value gets removed
NO
The frequency domain value stays undisturbed
Resize filter to match dimensions of Fourier Transform
Apply Filter to frequency Domain for Each colour Channel And Shift the frequency near the origin using FFT-Shift.
Combine filter colour Channels into a single Image using Inverse Fourier Transform
Display The FIltered Image after Combination
Calculate the Error Metrics Between Denoised Image and The Reference Image
Dispaly the Metrics and the Filtered image
Stop

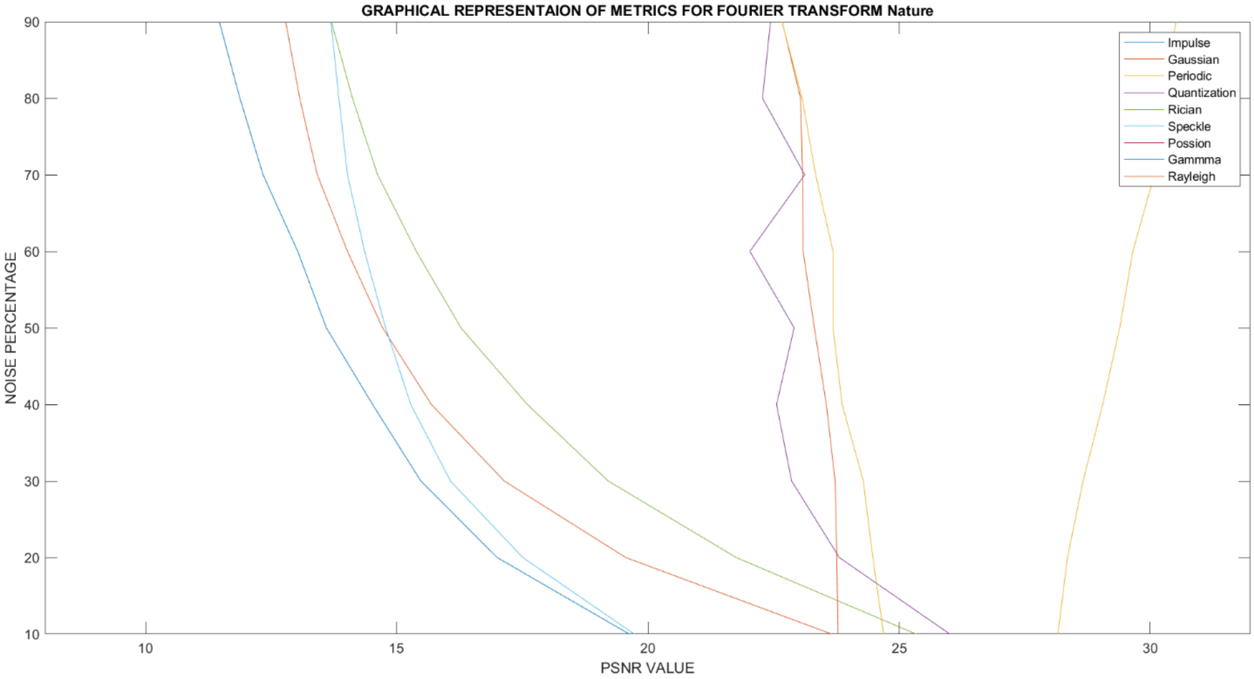
METRICS GRAPHICAL REPRESENTATION (NOISE PERCENT VS PSNR)

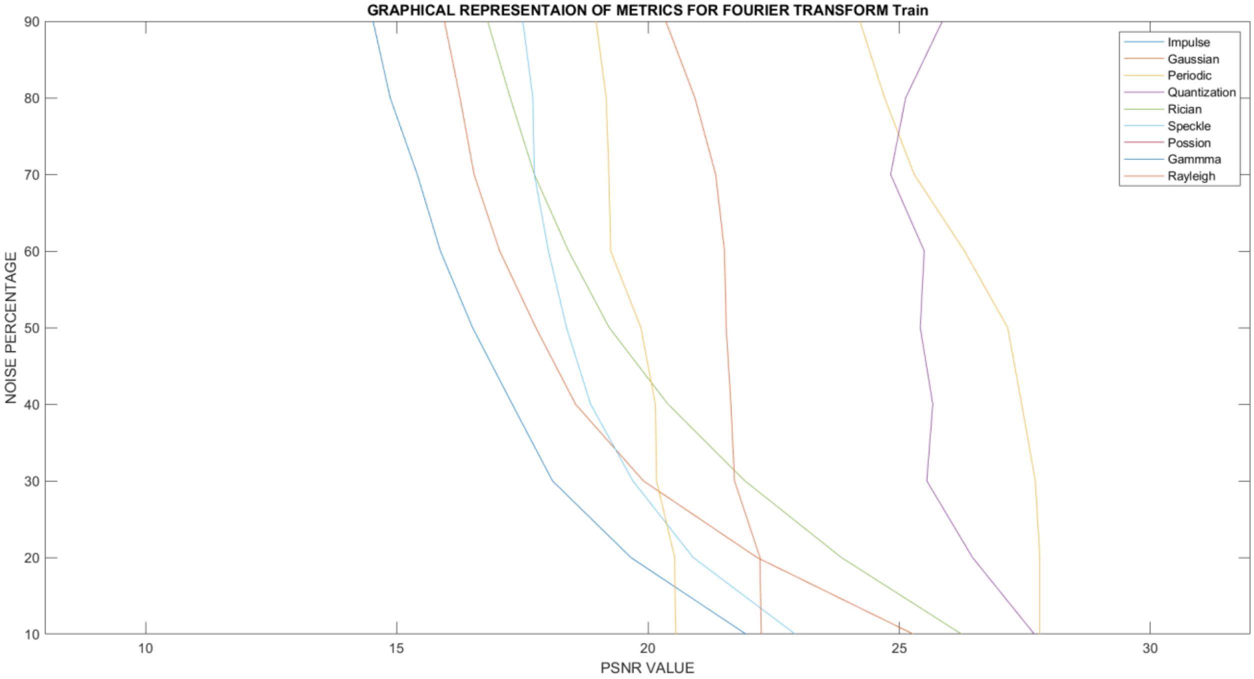
GRAPHICAL REPRESENTAION OF METRICS FOR FOURIER TRANSFORM



GRAPHICAL REPRESENTAION OF METRICS FOR FOURIER TRANSFORM Plants







CONCLUSION (BASED ON ABOVE GRAPHICAL RESULTS)

This technique works best on noise, such as Periodic noise, as it is a high-frequency noise that can be easily filtered out by a low-pass filter**.** It may not work as effectively on other types of noise, such as salt and pepper noise, as it is a non-linear noise that cannot be easily removed by a linear filter. It can also be seen that the nature of the graph for Gamma, Poisson , Rician , Impulse in all the four Graphs are of exponentially decreasing nature. The filter works best for periodic noise and slowly tends to show reduced PSNR values for as Noise increases specifically for periodic noise. While from the above graph , the PSNR values for Quantization noise slightly increases. Rayleigh and Gaussian Noise almost remain same or just slightly decreases in PSNR value irrespective of the Noise percentage.

RESEARCH PAPERS REFERENCES:

1) Fourier transform-based windowed minimum filter for reducing periodic noise from digital images.Justin Varghese1, Saudia Subash2, Nasser Tairan 11 Department of Computer Science, King Khalid University, Gregar, Abha, Asir 61411, KSA2 Centre for Information Technology and Engineering, Manonmaniam Sundaranar University, Tirunelveli, Tamil Nadu 627012, India. LINK:HERE

2) Brief review of image denoising techniques Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. LINK:HERE

3) Fourier block noise reduction: an adaptive filter for reducing Poisson noise in scintigraphic images Matthew J. Guy. LINK: HERE

4) Second International Symposium on Computer Vision and Internet(VisionNet'15).Image denoising by Fourier block processing and Weiner filtering, Naveen S., Aishwarya V.A. LINK:HERE

5) Image Denoising Using Wavelet Transform, Median Filter and Soft Thresholding Mersen Longkumer, Himanshu Gupta. LINK:HERE

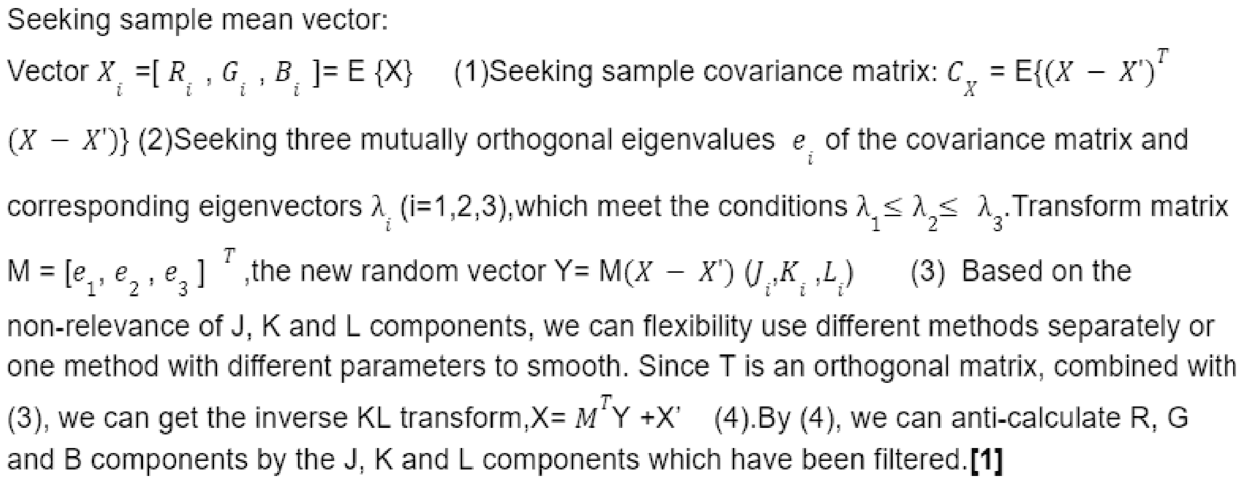
BRIEF THEORY AND EXPLANATION OF CODE OF K-L TRANSFORM-BASED FREQUENCY DOMAIN COLOURED IMAGE DENOISING FILTER.

THERORY AND METHODOLOGY:

The Karhunen-Loeve (K-L) transform, also known as the Principal Component Analysis (PCA) or Hotelling transform, is a mathematical technique for transforming a set of correlated variables into a set of linearly uncorrelated variables. The resulting variables are ordered so that the first variable has the highest possible variance, the second variable has the second-highest variance, and so on. The K-L transform is widely used in signal processing, data compression, and pattern recognition. The K-L transform is a frequency domain filter. It transforms an image from the spatial domain to the frequency domain, where it can be filtered by removing or attenuating certain frequency components before transforming it back to the spatial domain. The K-L transform uses the Discrete Cosine Transform (DCT) to convert the image to the frequency domain.

LITERATURE REVIEW:

The Karhunen-Loeve (K-L) transform, also known as the Principal Component Analysis (PCA) or Hotelling transform, is a mathematical technique for transforming a set of correlated variables into a set of linearly uncorrelated variables.. The K-L transform is a frequency domain filter. It transforms an image from the spatial domain to the frequency domain, where it can be filtered by removing or attenuating certain frequency components before transforming it back to the spatial domain. To denoise a noisy image, a logarithmic transformation is applied to convert multiplicative noise into additive noise. The image is segmented and the covariance matrix of each segment is calculated. The overall covariance matrix is obtained by averaging all segment covariances. Eigenvectors corresponding to the largest eigenvalues are selected to form a feature vector[2] . The proposed method consists of two stages. In the first stage, the K-L transform is applied to the noisy image to obtain the transform coefficients. In the second stage, a TV regularization method is applied to the transform coefficients. The TV regularization method enforces the sparsity of the transform coefficients and suppresses the noise. The filtered transform coefficients are then transformed back to the spatial domain to obtain the denoised image. The K-L transform uses the Discrete Cosine Transform (DCT) to convert the image to the frequency domain. KLT is based on the image's statistical properties. Its primary features are the ability to segregate information, get rid of connections between picture data, and emphasize the function of various objectives. It is frequently used to eliminate random noise while extracting coherent information from the seismic signal[5] **.**We could see the RGB image pixels as three-components of a three-dimensional random vector X



Seeking sample mean vector:

Vector X_i =[R_i , G_i , B_i]= E {X}   (1)Seeking sample covariance matrix: C_X = E{$(X - X')^T$ $(X - X')$} (2)Seeking three mutually orthogonal eigenvalues e_i of the covariance matrix and corresponding eigenvectors λ_i (i=1,2,3),which meet the conditions $\lambda_1 \leq \lambda_2 \leq \lambda_3$.Transform matrix M = $[e_1, e_2, e_3]^T$,the new random vector Y= M$(X - X')$ (J_i, K_i, L_i) (3) Based on the non-relevance of J, K and L components, we can flexibility use different methods separately or one method with different parameters to smooth. Since T is an orthogonal matrix, combined with (3), we can get the inverse KL transform,X= M^TY +X'   (4).By (4), we can anti-calculate R, G and B components by the J, K and L components which have been filtered.**[1]**

ALGORITHM

1.Load the noisy image and the reference image: Load the two images required for denoising.

2.Convert the noisy image to double precision: Converts the noisy image to double precision for better precision in calculations.

3.Separate the noisy image into its RGB components: Separates the three color channels (red, green, blue) of the noisy image into three individual matrices.

4.Compute the K-L transform of each component: Performs DCT on each color channel of the noisy image to compute the K-L transform of each component.

5.Set the threshold for denoising: Set the threshold value for denoising.

6.Apply soft thresholding to each component: Applies soft thresholding to each component of the K-L transformed noisy image to suppress the noise.

7.Compute the inverse K-L transform of each component: Performs IDCT on each color channel of the K-L transformed denoised image to obtain the denoised image in the RGB color space.

8.Combine the denoised RGB components into a single image: Combine the three denoised color channels into a single RGB image.

9.Compute the error parameters between the denoised image and the reference image: Compute the RMSE, MSE, SSIM, and PSNR values between the denoised image and the reference image.

10.Display the images: Displays the original noisy image, denoised image, and reference image side by side.

11. Display the error parameters: Displays the computed error parameters between the denoised image and the reference image.

PSEUDO CODE:

```
% Load the noisy image and the reference image
noisy_img = imread('impulse_20-plants.png');
ref_img = imread('plants.jpg');
ref_img = im2double(ref_img);

% Convert the noisy image to double precision
noisy_img = im2double(noisy_img);

% Separate the noisy image into its RGB components
R = noisy_img(:,:,1);
G = noisy_img(:,:,2);
B = noisy_img(:,:,3);

% Compute the K-L transform of each component
R_kl = dct2(R);
G_kl = dct2(G);
B_kl = dct2(B);

% Set the threshold for denoising
threshold = 0.3;

% Apply soft thresholding to each component
R_kl_denoised = sign(R_kl) .* max(abs(R_kl) - threshold, 0.01);
G_kl_denoised = sign(G_kl) .* max(abs(G_kl) - threshold, 0.01);
B_kl_denoised = sign(B_kl) .* max(abs(B_kl) - threshold, 0.01);

% Compute the inverse K-L transform of each component
R_denoised = idct2(R_kl_denoised);
G_denoised = idct2(G_kl_denoised);
B_denoised = idct2(B_kl_denoised);
```

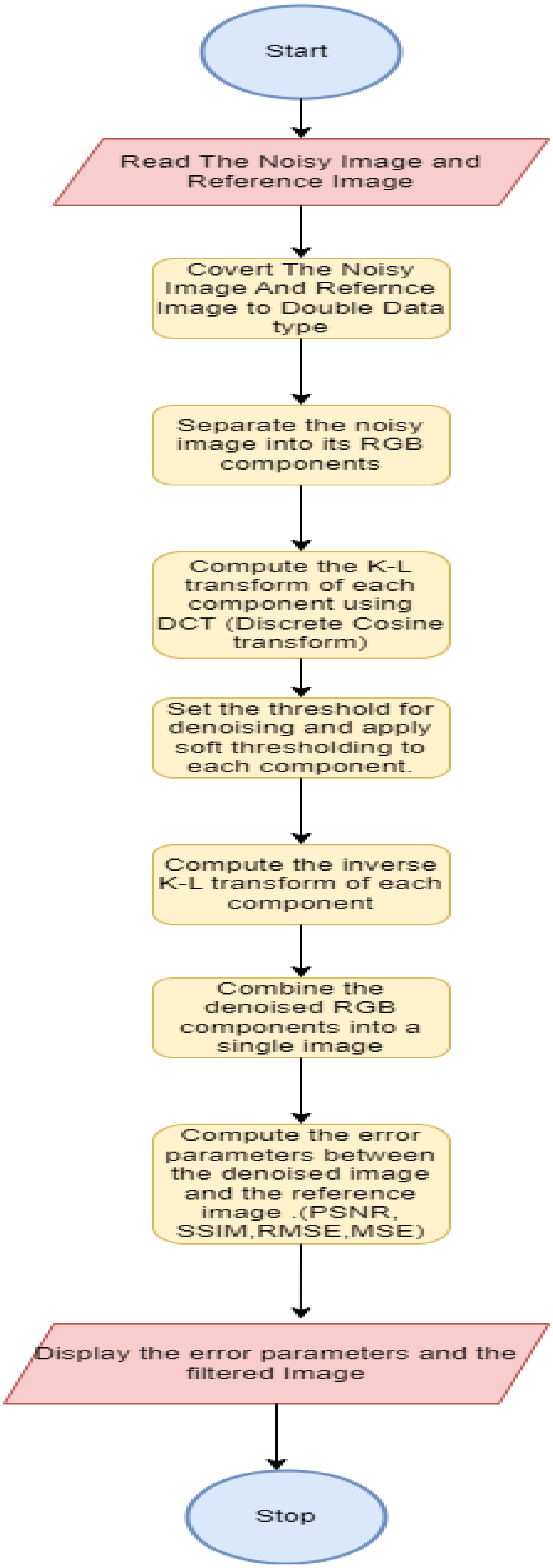
```
% Combine the denoised RGB components into a single image
denoised_img = cat(3, R_denoised, G_denoised, B_denoised);

% Compute the error parameters between the denoised image and the reference image
ref_img_resized = imresize(ref_img, [round(size(denoised_img, 1)), round(size(denoised_img, 2))]);
rmse = sqrt(mean((denoised_img(:) - im2double(ref_img_resized(:))).^2));
mse = mean((denoised_img(:) - im2double(ref_img_resized(:))).^2);
ref_img_resized = cast(ref_img_resized, class(denoised_img));
max_i = double(max(noisy_img(:)));
ssimval = ssim(denoised_img, ref_img_resized);
psnr = 20*log10(max_i/rmse);
% display the images
subplot(1,3,1);
imshow(noisy_img);
title('Noisy Image');

subplot(1,3,2);
imshow(denoised_img);
title('Denoised Image');

subplot(1,3,3);
imshow(ref_img);
title('Reference Image');

% Display the error parameters

fprintf('MSE: %.2f, RMSE: %.2f, PSNR: %.2f dB, SSIM: %.4f', mse, rmse, psnr, ssimval);
```

Variable and Parameter Description

VARIABLES			**DESCRIPTION**
noisy_img			The noisy image to be denoised.
ref_img			The reference image used to compare the denoised image
R, G, B			The RGB components of the noisy image
R_kl, G_kl, B_kl			The K-L transform of each RGB component

threshold			The threshold used for soft thresholding.
R_kl_denoised, G_kl_denoised, B_kl_denoised			The K-L transform of each RGB component after applying soft thresholding.
R_denoised, G_denoised, B_denoised			The denoised RGB components after applying inverse K-L transform.
denoised_img			The final denoised image after combining RGB components.
ref_img_resized			The resized reference image to match the size of the denoised image.
rmse			The root-mean-square error between the denoised image and the reference image.
mse			The mean square error between the denoised image and the reference image
max_i			The maximum pixel value of the noisy image.
ssimval			The structural similarity index between the denoised image and the reference image.
psnr			The peak signal-to-noise ratio between the denoised image and the reference image.

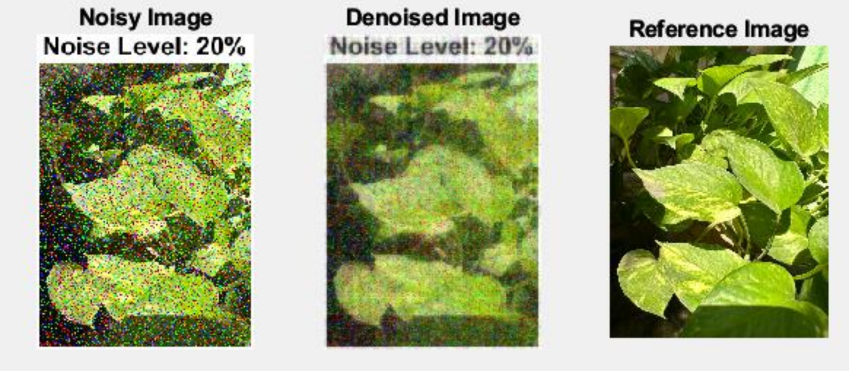
Flow Chart:


Start
Read The Noisy Image and Reference Image
Covert The Noisy Image And Refernce Image to Double Data type
Separate the noisy image into its RGB components
Compute the K-L transform of each component using DCT (Discrete Cosine transform)
Set the threshold for denoising and apply soft thresholding to each component.
Compute the inverse K-L transform of each component
Combine the denoised RGB components into a single image
Compute the error parameters between the denoised image and the reference image .(PSNR, SSIM,RMSE,MSE)
Display the error parameters and the filtered Image
Stop

FLOWCHART DESCRIPTION:

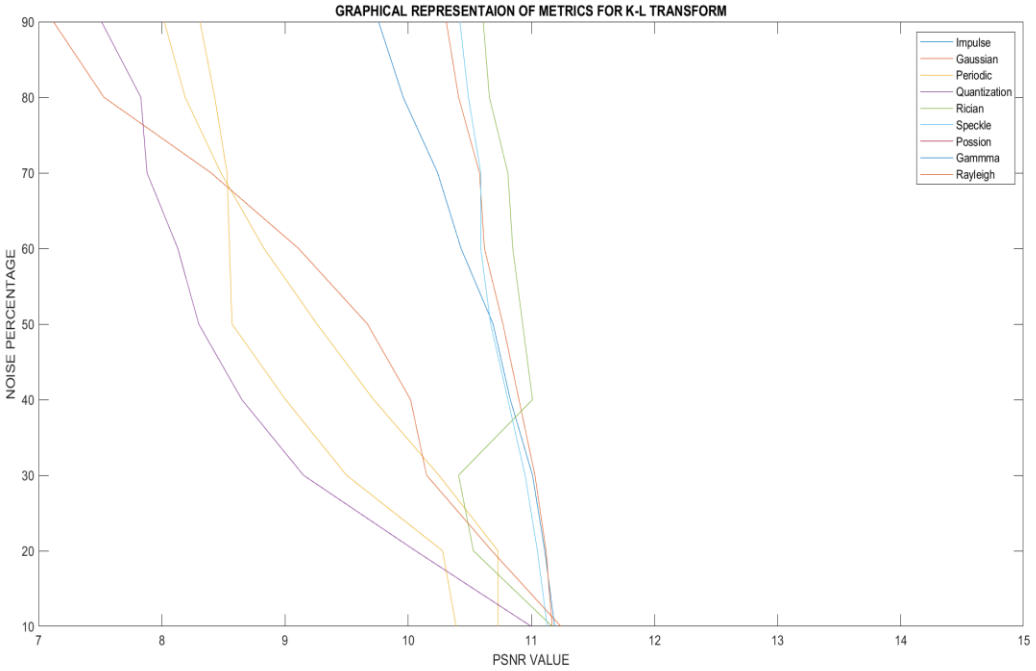
Load the noisy image and the reference image. Convert the noisy image to double precision. Separate the noisy image into its RGB components. Compute the K-L transform of each component. Set the threshold for denoising. Apply soft thresholding to each component. Compute the inverse K-L transform of each component. Combine the denoised RGB components into a single image. Compute the error parameters between the denoised image and the reference image. Display the noisy image, denoised image, and reference image Display the error parameters.

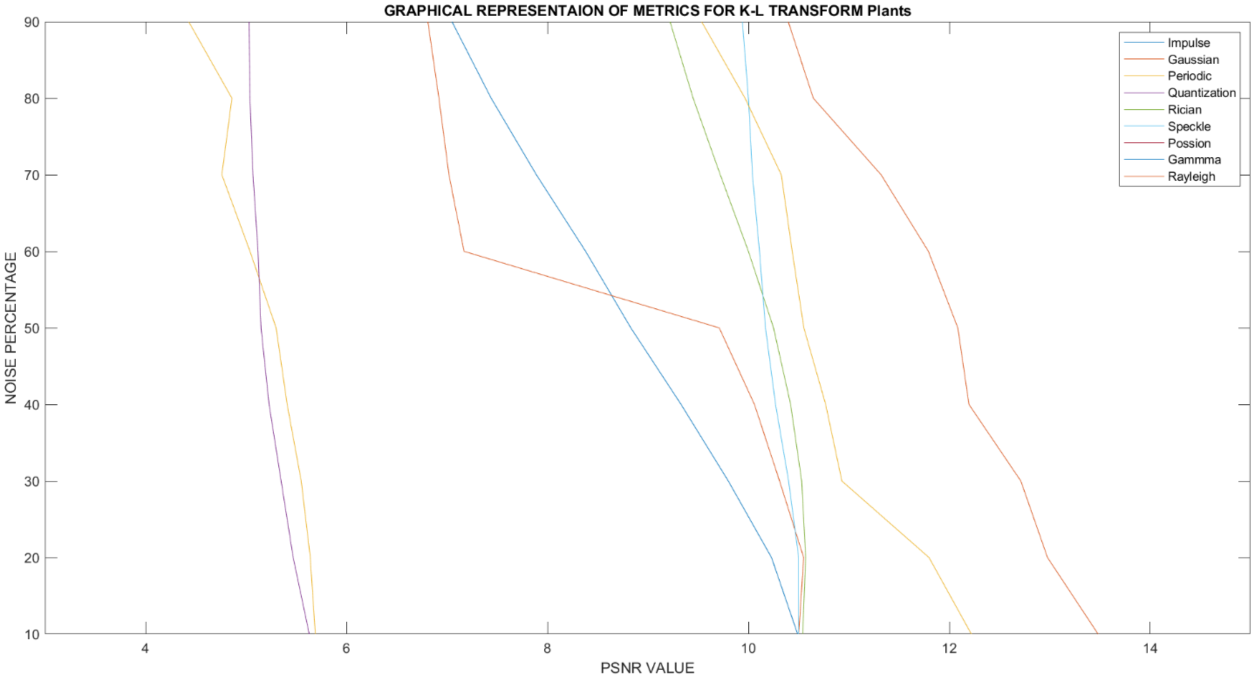
Sample output: (For 20% Impulse noise)

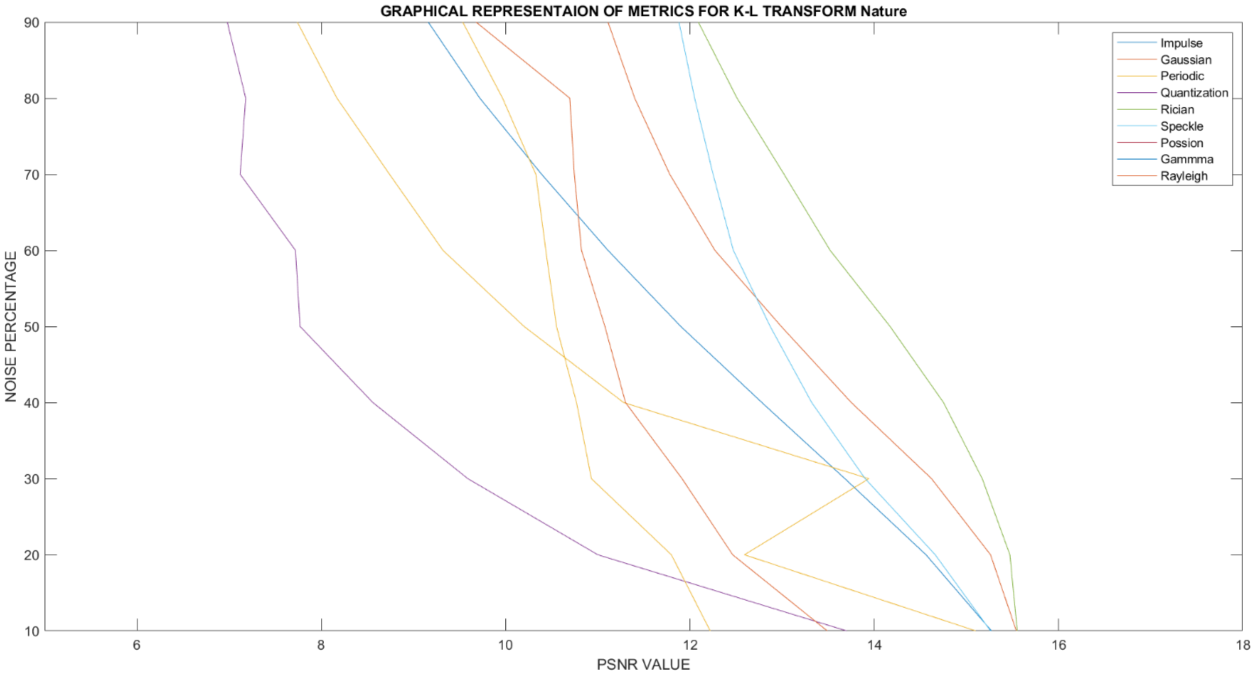


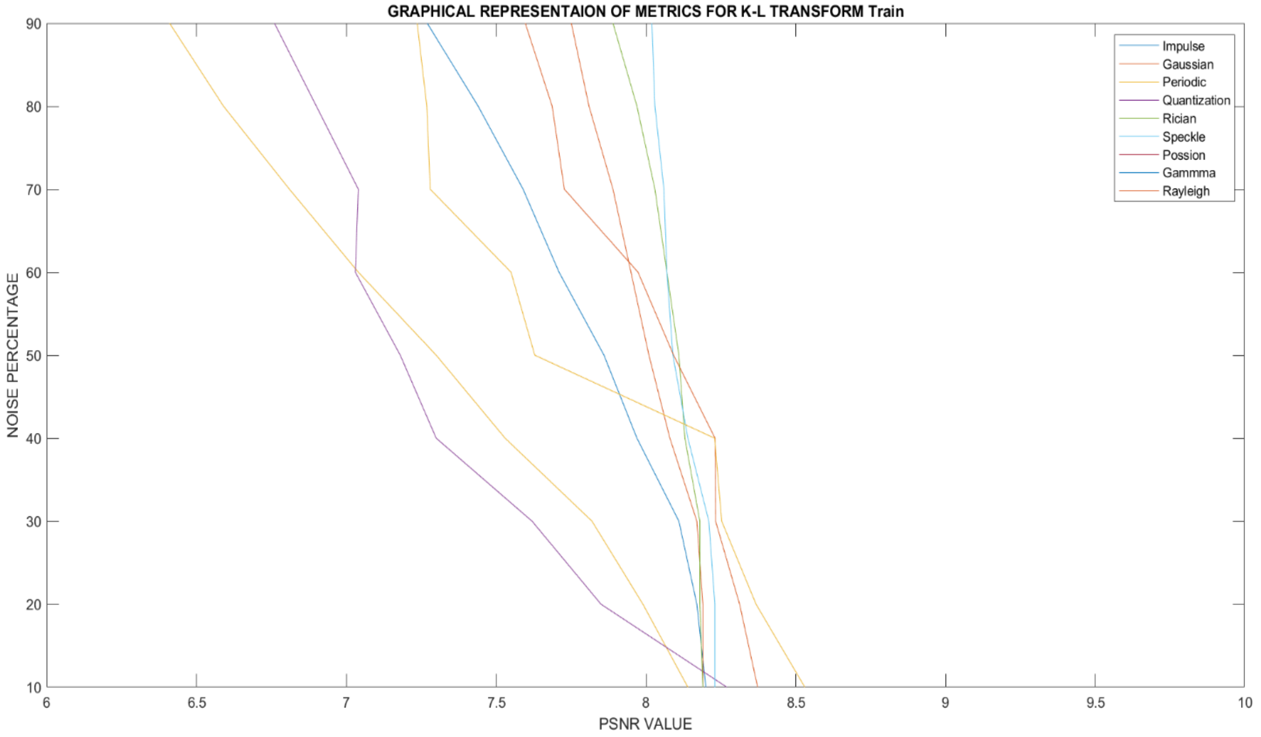
Error Metrics: MSE: 0.09, RMSE: 0.31, PSNR: 10.23 dB, SSIM: 0.4296

METRICS GRAPHICAL REPRESENTATION (NOISE PERCENT VS PSNR)




GRAPHICAL REPRESENTAION OF METRICS FOR K-L TRANSFORM Plants
NOISE PERCENTAGE
90
80
70
60
50
40
30
20
10
4
6
8
10
12
14
PSNR VALUE
Impulse
Gaussian
Periodic
Quantization
Rician
Speckle
Possion
Gammma
Rayleigh


GRAPHICAL REPRESENTAION OF METRICS FOR K-L TRANSFORM Nature
NOISE PERCENTAGE
90
80
70
60
50
40
30
20
10
6
8
10
12
14
16
18
PSNR VALUE
Impulse
Gaussian
Periodic
Quantization
Rician
Speckle
Possion
Gammma
Rayleigh



Conclusion:

From the above graphs we can conclude that K-L transform-based filters **work best on additive Gaussian noise**, which is a type of noise that is characterized by random variations in pixel values and has a Gaussian distribution. This is because the K-L transform is most effective at separating the signal from noise when the noise is Gaussian and uncorrelated. However, K-L transform-based filters may not be as effective on other types of noise, such as impulse noise or multiplicative noise, which have different statistical properties. This filter works the least for periodic and Quantization noises. The filter works almost works equally good for in case of Rician Noise, Impulse noise both. Though PSNR value for Rician Noise decreases with Increases in Noise , the PSNR value of Impulse Noise almost remains constant throughout the Graph.

RESEARCH PAPERS REFERENCES:

1) *A Method based on Wavelet Transform and Discrete K-L Transform for Color Image Filtering Xiaoqi Xue, Yu Zheng College of Science Tianjin Polytechnic University, TJPU Tianjin, China. LINK:HERE*

2) *ULTRASOUND IMAGE DE-NOISING THROUGH KARHUNEN-LOEVE (K-L) TRANSFORM WITH OVERLAPPING SEGMENTS Jawad F. Al-Asad, (Telephone: 773*

6930365, Email:jfalasad@uwm.edu) Alireza Moghadamjoo, (Telephone: 414 2295262, Email:reza@uwm.edu) Leslie Ying, (Telephone: 414 2295907, Email:leiying@uwm.edu) Department of Electrical Engineering and Computer Science. University of Wisconsin-Milwaukee USA. LINK:HERE

3) Performance Analysis of Image Denoising System for different levels of Wavelet decomposition S.Arivazhagan, S.Deivalakshmi, K.Kannan Department of Electronics and Communication Engineering, Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu, India. LINK:HERE

4) Block Based thresholding in Wavelet Domain for Denoising Ultrasound Medical Images P.V.V.Kishore MIEEE, A.S.C.S.Sastry, A.Kartheek, Sk.Harshad Mahatha Dept. of E.C.E, K.L. University, Green Fields, Vaddeswaram, Guntur DT, INDIA. LINK:HERE

5) Image Denoising Method based on Threshold, Wavelet Transform and Genetic Algorithm Yali Liu Shangluo University. LINK:HERE