MA5710 - Assignment 3

Abhinav T K, MA23M002

September 30, 2023

1. **Implementation of EED model.**

   Code: **eed.m**

```

function g = eed(f,scale, k, stepsize,nosteps,verbose,ip)
% Edge Enhancing diffusion

if verbose
    figure(verbose);
    subplot(1,2,1);
    imshow(f);
    title('Original Image');
    drawnow;
end

% Diffusion tensor D = D(delu) = (v1, v2)diag(l1, l2)(v1T, v2T)
% D = G^T.diag(c1,c2).G ; G = (gx, gy)
g = f;
for i = 1:nosteps
    %Calculating gx, gy and gw at given scale for diffusion tensor;
    gx = gD( g, scale, 1, 0 );      % Computing the gradient
    gy = gD( g, scale, 0, 1 );

    grad2 = gx.*gx+gy.*gy;
    gw = sqrt(grad2);

    %Calculating c1 and c2
    c2 = exp(-(gw/k).^2);
    c1 = (1/5)*c2;

    %Calculating diffusion tensor components for tnldStep function
    a = (c1.*gx.^2 + c2.*gy.^2)./grad2;
    b = (c2-c1).*gx.*gy./grad2;
    c = (c1.*gy.^2 + c2.*gx.^2)./grad2;

    g = g + stepsize * tnldStep( g, a, b, c, ip);

    %If verbose show edge enhanced diffusion
    if verbose
        figure(verbose);
        subplot(1,2,2);
        imshow(g);
        title('Edge Enhancing Diffusion');
        drawnow;
    end
end
```

Code: **eedtest.m**

```
% Test eed model
clc;
clear;

a=imread('cameraman.tif'); % reading the image
a=im2double(a); % normalizing the instensity values to lie between 0 and 1

ref=a;
ad=imnoise(a,'gaussian',0.01); % adding Gaussian noise of mean zero and variance
    0.01
timestep=0.2; % timestep size used in numerical approximation
Niter=60; % number of iterations

b = eed(ad,1,0.05,timestep,Niter,1,1);      % Edge Enhancing function

% Arguments
% 1 is the noisy image,
% 2 is the scale - (variance in Gaussian),
% 3 is the lambda value = contrast parameter,
% 4 is the timestep size, 5 is the no of iterations,
% 6 is the value to show the plot,
% 7 is the w value used in numerical approximation
% 8 corresponding to choice of the numerical scheme.
```

Output:

Figure 1: Output of eed model for contrast parameter - 0.05

2. Take a standard image(for example, cameraman image), perform Linear diffusion, PMC, EED and compare the respective PSNR values.

Code: **q2.m**

```
% Linear diffusion, PMC, EED for cameraman image and their outputs
clc;
clear;

a=imread('cameraman.tif'); % reading the image
a=im2double(a); % normalizing the instensity values to lie between o and 1

ref=a;
ad=imnoise(a,'gaussian',0.01); % adding Gaussian noise of mean zero and variance 0.01
timestep=0.2; % timestep size used in numerical approximation
Niter=60; % number of iterations

alpha=2.7;              % Used in Numerical approximation of pmc
w= exp(4*alpha/9);      % Used in Numerical approximation of pmc

output_linear = imgaussfilt(ad,1);    %Filter the image with Linear Diffusion (usingGaussian filter) with sigma = 1
output_pmc =pmc(ad,ref,0.5,timestep,Niter,0,w,1); %Filter the image with Perona Malik diffusion with lambda = 0.5
output_eed = eed(ad,1,0.05,timestep,Niter,0,1);   % Filter the image with Edge Enhancing Diffusion with lambda = 0.05

% PSNR - Metric to understand the quality of cleaning
psnr_linear = psnr(output_linear,ad);        % PSNR value for Linear diffusion with sigma = 1
psnr_pmc = psnr(output_pmc,ad);              % PSNR value for Perona Malik diffusion with lambda = 0.5
psnr_eed = psnr(output_eed,ad);              % PSNR value for Edge Enhancing diffusion with lambda = 0.05

figure(1);
montage({ad,output_linear});    % Compare noise and output image of Linear diffusion
title('Noisy Image Vs. Linear diffusion (sigma = 1) ');

figure(2);
montage({ad,output_pmc});    % Compare noise and output image of PMC
title('Noisy Image Vs. Perona Malik diffusion (lambda = 0.5) ');

figure(3);
montage({ad,output_eed});    % Compare noise and output image of EED
title('Noisy Image Vs. Edge Enhancing diffusion (lambda = 0.05) ');

fprintf('PSNR Value for Linear diffusion with (sigma = 1) = %.2f \n',psnr_linear);
fprintf('PSNR Value for PMC with (lambda = 0.5) = %.2f \n',psnr_pmc);
fprintf('PSNR Value for EED with (lambda = 0.05) = %.2f \n',psnr_eed);
```

Output:

PSNR Value for Linear diffusion with (sigma = 1) = 20.36
PSNR Value for PMC with (lambda = 0.5) = 21.13
PSNR Value for EED with (lambda = 0.05) = 19.84

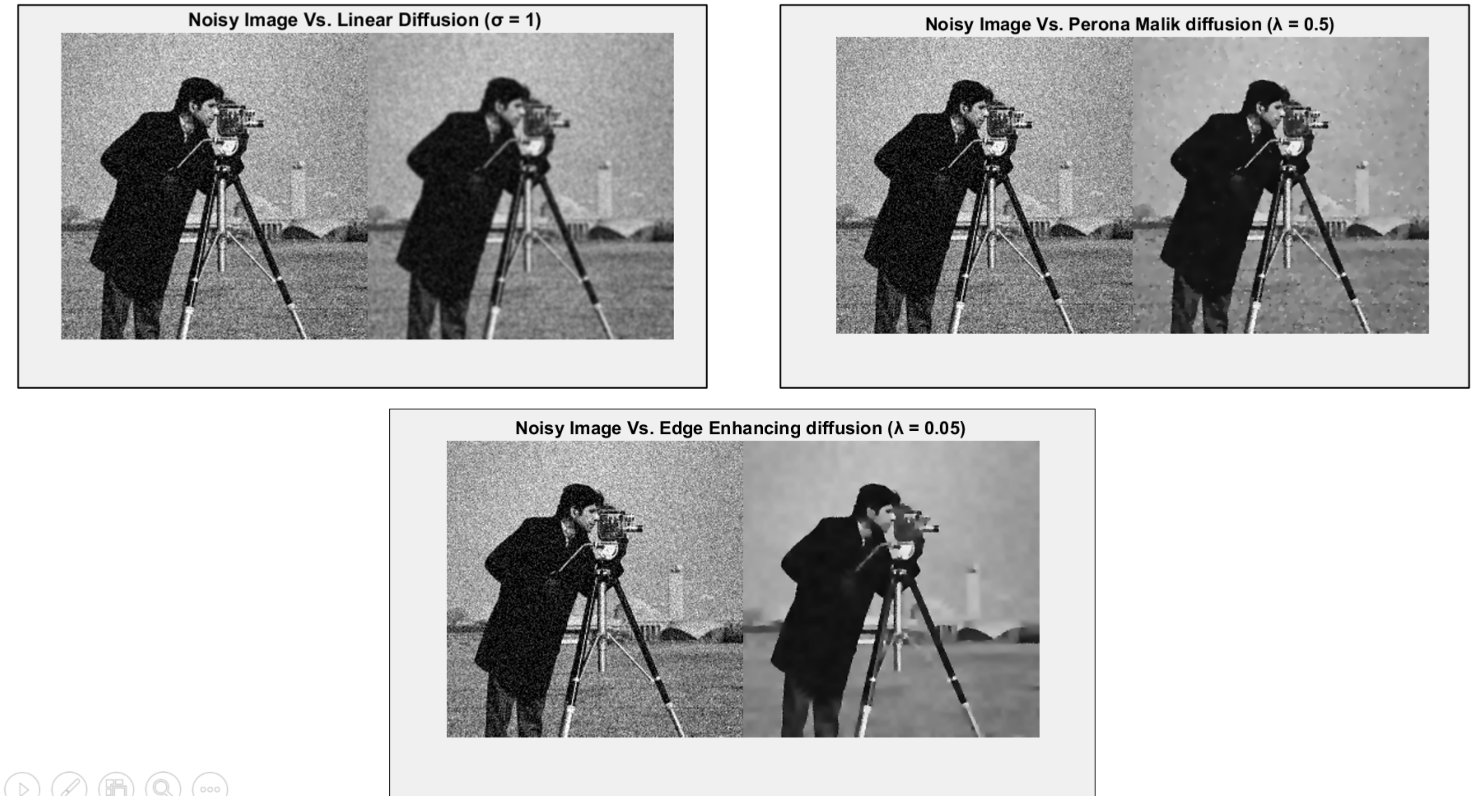

Figure 2: Comparison of all 3 models

3. Take 3 or 4 different images, add different noises to them(Gaussian noise, Salt and Pepper noise, speckle noise and poisson noise with different variances), treat each case as different input image, apply LD, PMC and EED, report respective PSNR values. For better interpretation of your outputs, compare the three models on variance vs PSNR plots for each noise and each image.

Code: q3.m

```
% A3-q3
clc;
clear;

% Importing 5 images
img1 = im2double(imread('cameraman.tif')); % reading the image and normalizing the instensity values b/w 0 and 1
img2 = im2double(imread('tomo.jpg'));
img3 = im2double(im2gray(imread('triangle.jpg'))); % converting color img to greyscale
img4 = im2double(im2gray(imread('onion.png')));
img5 = im2double(imread('eight.tif'));

n_img = 5; % No. of images
n_noise = 10;   % No. of noisy images for each image

% Declare a cellArray of size 5x10 to store noisy images of each image
cellArray = cell(n_img,n_noise);

% Adding Gaussian noise, Poisson, Salt & Pepper noise and speckle noise with different variances

cellArray(1,1) = {img1};                                    % original image1
cellArray(1,2) = {imnoise(img1, 'gaussian',0.01)};  % image with noise
cellArray(1,3) = {imnoise(img1, 'gaussian',0.1)};
cellArray(1,4) = {imnoise(img1, 'gaussian',0.5)};
cellArray(1,5) = {imnoise(img1, 'poisson')};
cellArray(1,6) = {imnoise(img1, 'salt & pepper',0.01)};
cellArray(1,7) = {imnoise(img1, 'salt & pepper',0.05)};
cellArray(1,8) = {imnoise(img1, 'salt & pepper',0.1)};
cellArray(1,9) = {imnoise(img1, 'speckle',0.01)};
cellArray(1,10) = {imnoise(img1, 'speckle',0.1)};

cellArray(2,1) = {img2};                                    % original image2
cellArray(2,2) = {imnoise(img2, 'gaussian',0.01)};  % image with noise
cellArray(2,3) = {imnoise(img2, 'gaussian',0.1)};
cellArray(2,4) = {imnoise(img2, 'gaussian',0.5)};
cellArray(2,5) = {imnoise(img2, 'poisson')};
cellArray(2,6) = {imnoise(img2, 'salt & pepper',0.01)};
cellArray(2,7) = {imnoise(img2, 'salt & pepper',0.05)};
cellArray(2,8) = {imnoise(img2, 'salt & pepper',0.1)};
cellArray(2,9) = {imnoise(img2, 'speckle',0.01)};
cellArray(2,10) = {imnoise(img2, 'speckle',0.1)};

cellArray(3,1) = {img3};                                    % original image3
cellArray(3,2) = {imnoise(img3, 'gaussian',0.01)};  % image with noise
cellArray(3,3) = {imnoise(img3, 'gaussian',0.1)};
cellArray(3,4) = {imnoise(img3, 'gaussian',0.5)};
cellArray(3,5) = {imnoise(img3, 'poisson')};
cellArray(3,6) = {imnoise(img3, 'salt & pepper',0.01)};
```

```
cellArray(3,7) = {imnoise(img3, 'salt & pepper',0.05)};
cellArray(3,8) = {imnoise(img3, 'salt & pepper',0.1)};
cellArray(3,9) = {imnoise(img3, 'speckle',0.01)};
cellArray(3,10) = {imnoise(img3, 'speckle',0.1)};

cellArray(4,1) = {img4};                              % original image4
cellArray(4,2) = {imnoise(img4, 'gaussian',0.01)};   % image with noise
cellArray(4,3) = {imnoise(img4, 'gaussian',0.1)};
cellArray(4,4) = {imnoise(img4, 'gaussian',0.5)};
cellArray(4,5) = {imnoise(img4, 'poisson')};
cellArray(4,6) = {imnoise(img4, 'salt & pepper',0.01)};
cellArray(4,7) = {imnoise(img4, 'salt & pepper',0.05)};
cellArray(4,8) = {imnoise(img4, 'salt & pepper',0.1)};
cellArray(4,9) = {imnoise(img4, 'speckle',0.01)};
cellArray(4,10) = {imnoise(img4, 'speckle',0.1)};

cellArray(5,1) = {img5};                              % original image5
cellArray(5,2) = {imnoise(img5, 'gaussian',0.01)};   % image with noise
cellArray(5,3) = {imnoise(img5, 'gaussian',0.1)};
cellArray(5,4) = {imnoise(img5, 'gaussian',0.5)};
cellArray(5,5) = {imnoise(img5, 'poisson')};
cellArray(5,6) = {imnoise(img5, 'salt & pepper',0.01)};
cellArray(5,7) = {imnoise(img5, 'salt & pepper',0.05)};
cellArray(5,8) = {imnoise(img5, 'salt & pepper',0.1)};
cellArray(5,9) = {imnoise(img5, 'speckle',0.01)};
cellArray(5,10) = {imnoise(img5, 'speckle',0.1)};

% Initialize psnr array to store psnr values
psnr_array = zeros(5,10,3);

% Apply LD, PMC and EED for each noisy images and report respective PSNR values

timestep = 0.2; % timestep size used in numerical approximation
Niter = 60; % number of iterations

alpha=2.7;              % Used in Numerical approximation of pmc
w= exp(4*alpha/9);      % Used in Numerical approximation of pmc

for i = 1:n_img
    for j = 1:n_noise   %
        output_linear = imgaussfilt(cellArray{i,j},0.5);    %Filter the image with Linear Diffusion ( usingGaussian filter) with sigma = 0.5
        output_pmc =pmc(cellArray{i,j},cellArray{i,1},0.05,timestep,Niter,0,w,1); % Filter the image with Perona Malik diffusion with lambda = 0.05
        output_eed = eed(cellArray{i,j},1,0.05,timestep,Niter,0,1);   % Filter the image with Edge Enhancing Diffusion with lambda = 0.05
        psnr_array(i,j,1) = psnr(output_linear,cellArray{i,1});         % PSNR value for Linear diffusion with sigma = 0.5
        psnr_array(i,j,2) = psnr(output_pmc,cellArray{i,1});            % PSNR value for Perona Malik diffusion with lambda = 0.05
        psnr_array(i,j,3) = psnr(output_eed,cellArray{i,1});          % PSNR value for Edge Enhancing diffusion with lambda = 0.05

    end
end
```

Output:
PSNR values for each model and image with noise is stored in psnr_array .

Code: **q3_PSNRvVar.m**

```
% A3q3 - PSNR vs variance
clc;
clear;

% Importing 5 images
img1 = im2double(imread('cameraman.tif')); % reading the image and normalizing the instensity values b/w 0 and 1
img2 = im2double(imread('tomo.jpg'));
img3 = im2double(im2gray(imread('triangle.jpg'))); % converting color img to greyscale
img4 = im2double(im2gray(imread('onion.png')));
img5 = im2double(imread('eight.tif'));


timestep = 0.2; % timestep size used in numerical approximation
Niter = 60; % number of iterations

alpha=2.7;               % Used in Numerical approximation of pmc
w= exp(4*alpha/9);       % Used in Numerical approximation of pmc

n = 100;                     % No. of variance values
var = logspace(-5,0,n)';     % Variance for the noise
ldpsnr = zeros(1,n);
pmcpsnr = zeros(1,n);
eedpsnr = zeros(1,n);
% Apply LD, PMC and EED for each noisy images and report respective PSNR values

%% Change img and noise for different conditions
img = img1;                  % img = img1, img2, img3, img4, img5
noise = "gaussian";          % noise type = gaussian, salt & pepper, speckle

for i = 1:n
    output_linear = imgaussfilt(imnoise(img, noise,var(i)),0.5);    %Filter the image with Linear Diffusion ( usingGaussian filter) with sigma = 0.5
    ldpsnr(i) = psnr(output_linear,img);

    output_pmc =pmc(imnoise(img, noise,var(i)),img,0.05,timestep,Niter,0,w,1); % Filter the image with Perona Malik diffusion with lambda = 0.05
    pmcpsnr(i) = psnr(output_pmc,img);

    output_eed = eed(imnoise(img, noise,var(i)),1,0.05,timestep,Niter,0,1);   % Filter the image with Edge Enhancing Diffusion with lambda = 0.05
    eedpsnr(i) = psnr(output_eed,img);
end

figure(1);
plot(var, ldpsnr, 'b-', 'LineWidth', 1);
hold on;
plot(var, pmcpsnr, 'r-', 'LineWidth', 1);
hold on;
plot(var, eedpsnr, 'g-', 'LineWidth', 1);

xlabel('Variance in noise');
ylabel('PSNR');
title(['Plot of Variance vs PSNR for noise: ', noise]);
legend('LD', 'PMC', 'EED');
```

Output:
Given output is for cameraman image and Gaussian noise.
images folder contains output of all combinations of image and noise.

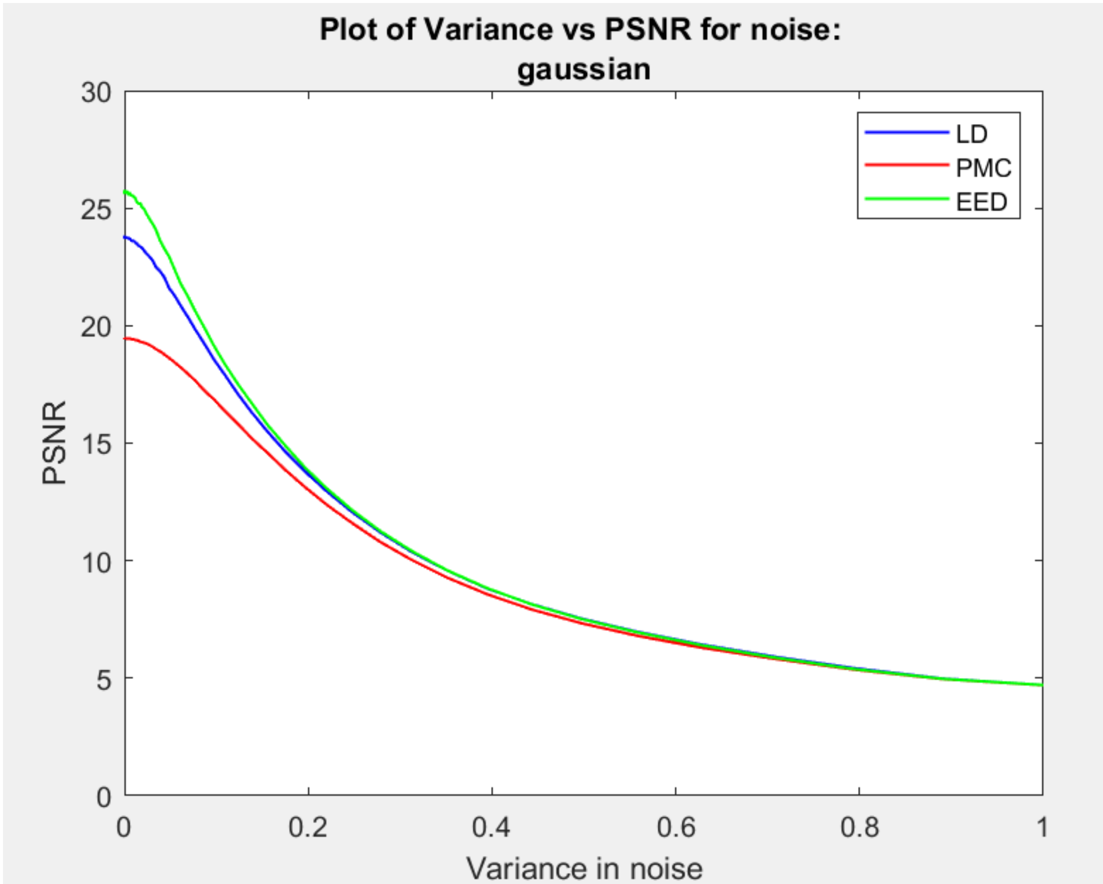

Figure 3: PSNR vs Variance of Gaussian noise for cameraman image

4. Figure out how sensitive are the model parameters? (Hint: For example, You can compare the three models on lambda, contrast parameter vs PSNR plot)

Code: **q4_PSNRvlambda.m**

```
% A3q4 - PSNR vs Contrast parameter (lambda)
clc;
clear;

% Importing 5 images
img1 = im2double(imread('cameraman.tif')); % reading the image and normalizing the
    instensity values b/w 0 and 1
img2 = im2double(imread('tomo.jpg'));
img3 = im2double(im2gray(imread('triangle.jpg'))); % converting color img to
    greyscale
img4 = im2double(im2gray(imread('onion.png')));
img5 = im2double(imread('eight.tif'));


timestep = 0.2; % timestep size used in numerical approximation
Niter = 60; % number of iterations

alpha=2.7;              % Used in Numerical approximation of pmc
w= exp(4*alpha/9);      % Used in Numerical approximation of pmc

n = 100;                    % No. of variance values
lambda = logspace(-5,0,n)';    % Lambda - contasrt parameter
ldpsnr = zeros(1,n);
pmcpsnr = zeros(1,n);
eedpsnr = zeros(1,n);
% Apply LD, PMC and EED for each noisy images and report respective PSNR values

%% Change img and noise for different conditions
img = img1;                 % img = img1, img2, img3, img4, img5
noise = "speckle";          % noise type = gaussian, salt & pepper, speckle, poisson

for i = 1:n
    output_linear = imgaussfilt(imnoise(img, noise),0.5);    %Filter the image with
        Linear Diffusion ( usingGaussian filter) with sigma = 0.5
    ldpsnr(i) = psnr(output_linear,img);

    output_pmc =pmc(imnoise(img, noise),img,lambda(i),timestep,Niter,0,w,1); %Filter
        the image with Perona Malik diffusion for different lambda
    pmcpsnr(i) = psnr(output_pmc,img);

    output_eed = eed(imnoise(img, noise),1,lambda(i),timestep,Niter,0,1);   % Filter
        the image with Edge Enhancing Diffusion for different lambda
    pmcpsnr(i) = psnr(output_pmc,img);
    eedpsnr(i) = psnr(output_eed,img);
end

figure(1);
plot(lambda, ldpsnr, 'b-', 'LineWidth', 1);
hold on;
plot(lambda, pmcpsnr, 'r-', 'LineWidth', 1);
hold on;
plot(lambda, eedpsnr, 'g-', 'LineWidth', 1);
hold off;
```

```
xlabel('Contrast parameter, Lambda');
ylabel('PSNR');
title(['Plot of Lambda vs PSNR for noise: ', noise]);
legend('LD', 'PMC', 'EED');
```

Output:
Output images of different combinations of image and noise is in **images folder**.

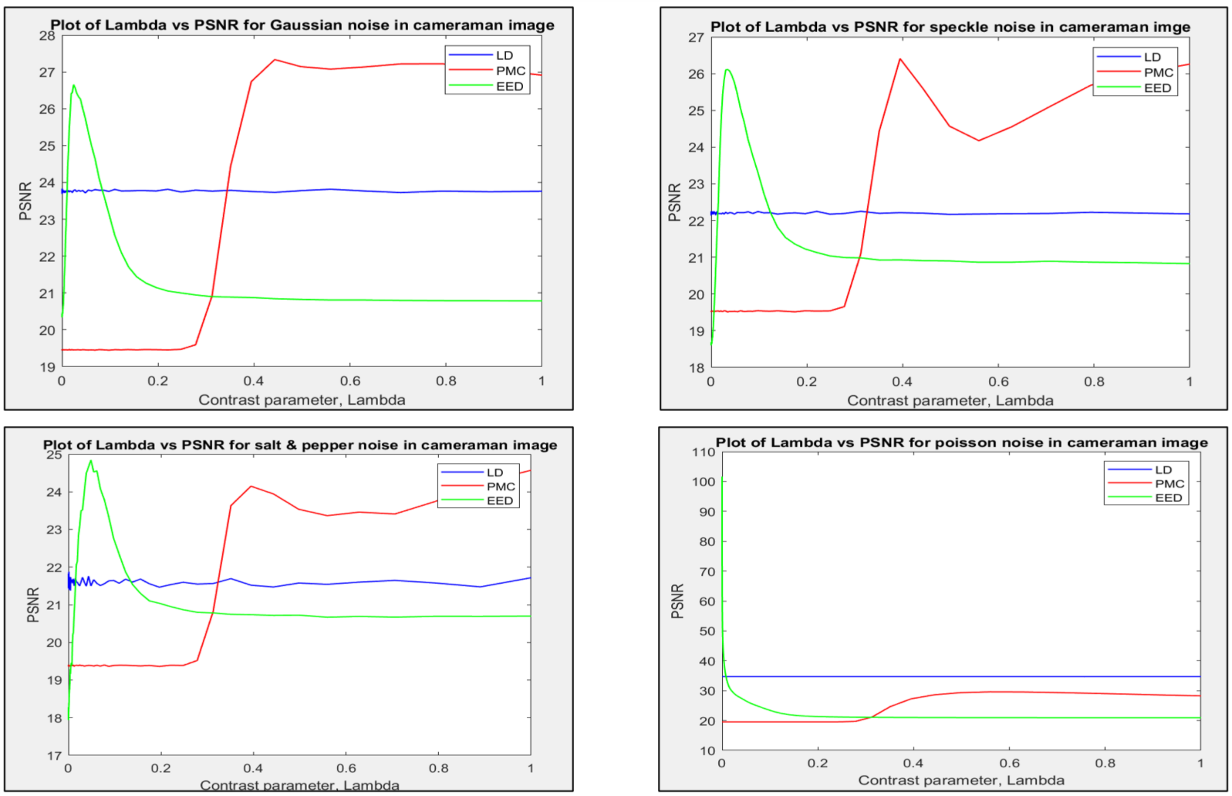

Figure 4: PSNR vs Lambda for cameraman image with different noises

Code: **eedtest_sigmaVlambda.m**

```
%% EED Model- contrast parameter lambda & smoothing parameter sigma relation
clc;
clear;

a=imread('cameraman.tif'); % reading the image
a=im2double(a); % normalizing the instensity values to lie between o and 1

ref=a;
ad=imnoise(a,'gaussian',0.01); % adding Gaussian noise of mean zero and variance
    0.01
timestep=0.2; % timestep size used in numerical approximation
Niter=60; % number of iterations

sigma = [0.1, 1, 3];                        % Sigma - smoothing parameter
lambda = [0.001, 0.01, 0.1, 1, 10];         % Lambda - contrast parameter

subplotno = 0;
for i=1:3
    for j=1:5
        b = eed(ad,sigma(i),lambda(j),timestep,Niter,0,1);      % Edge Enhancing
            function
        % Display the image in a subplot
         subplotno = subplotno+1;
         subplot(3, 5, subplotno); % 3 rows, 5 columns of subplots
         imshow(b);
         title(['sigma = ',num2str(sigma(i)),',  lambda = ', num2str(lambda(j))]);
    end
end
sgtitle('EED Model- contrast parameter lambda & smoothing parameter sigma relation')

% Arguments
% 1 is the noisy image,
% 2 is the scale - (variance in Gaussian),
% 3 is the lambda value = contrast parameter,
% 4 is the timestep size, 5 is the no of iterations,
% 6 is the value to show the plot,
% 7 is the w value used in numerical approximation
% 8 corresponding to choice of the numerical scheme.
```

Output:

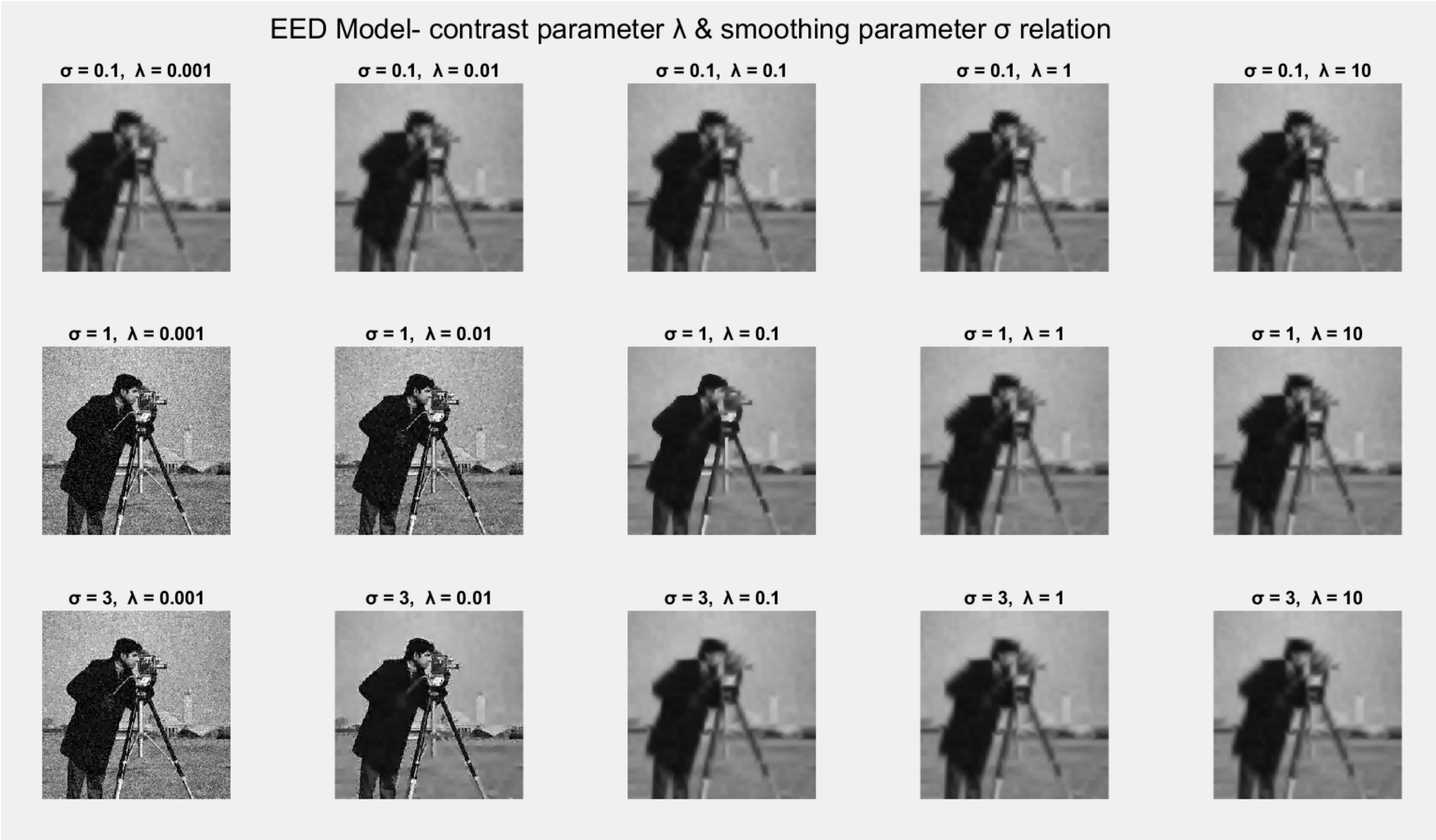

Figure 5: Relation b/w sigma and lambda for EED model

5. From the above findings, can you comment anything on the best stopping time for the respective models?

   The best stopping time is the one that gives a good balance between noise reduction and edge preservation for our input images. It can be found by experimenting the above models with different stopping times and finding their PSNR values to get an idea. Stopping time corresponding to higher PSNR value is can be considered a good stopping time.