```
clc;
close all;
clear all;

%taking image
name='input.jpg';
%img1 = bilateralFilter(name);
img = double(imread(name))/255;

%setting parameters
output='smoothing.jpg';
sig=5;
sectors=8;
quan=5;
epsilon = 10^-3;
[size1, size2, N] = size(img);

%this part is used for creating gaussian kernel
[x,y] = meshgrid(linspace(-size2/2, size2/2, size2),
linspace(size1/2, -size1/2, size1));
gaussianKernel = exp( -(x.^2+y.^2) / (2*sig^2) );

%this part converts each channel of image to fourier
domain.
for i = 1 : N
    imW(:,:,i)  = fft2(img(:,:,i));
    im2W(:,:,i) = fft2(img(:,:,i).^2);
end

%this part creates zero matrices
num = zeros(size1,size2,N);
den = zeros(size1,size2);

for i = 0 : sectors-1

%this part creates cutting functions and multiplies it with
gaussian kernel
%this operation creates weighting functions
    G = smoothgaux(sector(size1,size2, i*2*pi/sectors,
pi/sectors), 1, 2.5) .* gaussianKernel;
    G = G / sum(G(:));

%this part converts weighting functions to fourier domain
    G = fft2(G);

    S = zeros(size1,size2);

%this part calculates mean and standart deviation values
%this operation made in fourier domain and after that it
converted back
    for k = 1 : N
```

```
        m(:,:,k) = ifft2(G .* imW(:,:,k));
        S = S +    ifft2(G .* im2W(:,:,k)) - m(:,:,k).^2;
    end

%this part applies same operation for each channel in color
images
    S = (S+epsilon).^(-quan/2);
    den = den + S;
    for k = 1 : N
        num(:,:,k) = num(:,:,k) + m(:,:,k).*S;
    end

end

%this part creates output image
for k = 1 : N
    y(:,:,k) = fftshift(num(:,:,k) ./ den);
end

figure();imshow(y);
imwrite(y, output);

function y = smoothgaux(img, sigma, n)

U = ceil(sigma*n);
gauss = exp(-[-U:U].^2/2/sigma^2);
gauss = gauss/sum(gauss);

x = zeros(size(img));
for i = -U:U
    x = x + circshift(img, [i,0,0]) *gauss(i+U+1);
end

y = zeros(size(img));
for i = -U:U
    y = y + circshift(x, [0,i,0]) *gauss(i+U+1);
end
end

%this part is used for creating sectors
function S = sector(size1, size2, a, wd)

[x,y] = meshgrid(linspace(-size2/2, size2/2, size2),
linspace(size1/2, -size1/2, size1));

S = (x*cos(a-wd+pi/2) + y*sin(a-wd+pi/2) > 0) .*
(x*cos(a+wd+pi/2) + y*sin(a+wd+pi/2) <= 0);
end
```

```
clc;
close all;
clear all;
name='input.jpg';
%img1 = bilateralFilter(name);
img = double(imread(name))/255;
figure();imshow(img);title('main');
A = imread('smoothing','jpg');
%L = fspecial('average',[25,25]);
B = fspecial('average',[5,5]);
imshow(A);title('original image');
size(A);
C = rgb2gray(A);
figure();imshow(C);title('grey scale image');
size(C);
A = im2double(C);
%size(A)
D = conv2(A,B,'same');
figure();imshow(D);title('convolved image');
Q = A-D;
figure();imshow(Q);title('edges before normalized');
S = size(Q);
for i = 1:S(1)
for j = 1:S(2)
    if Q(i,j)> 0.03
    Q(i,j) = 0;
    else
    Q(i,j) = 1;
    end
end
end
V = fspecial('average',[1,1]);
G = conv2(Q,V,'same');
figure();
size(G);
imshow(G);title('Final image');
output='edge.jpg';
imwrite(G, output);
```

```
clc;
close all;
clear all;
name='input.jpg';
%img1 = bilateralFilter(name);
figure();imshow(name);title('main');
A = imread('smoothing','jpg');
B = fspecial('average',[5,5]);
figure();imshow(A);title('After normal Smoothing with
average');
edge = imread('edge','jpg');
figure();imshow(edge);title('edges');


rgbimage=A;
e= edge;
mask = e;
b = rgbimage;
[m,n]=size(b);

c = imfuse(b,mask,'blend','Scaling','joint');
%output (:,:,3) = rgbimage(:,:,3) .* mask;
%output (:,:,2) = rgbimage(:,:,2) .* mask;
%output (:,:,1) = rgbimage(:,:,1) .* mask;
%
%bIn3Dims = repmat(b,1,1,3);
%maskedRgbImage = bsxfun(@times, rgbimage, cast(mask,
'like', rgbimage));
% for i=1:m
 %        if (b(i)==0)
  %   maskedRgbImage(i,:)=rgbimage(i,:);
   %       end
 %end
  figure();imshow(c);
  n = 1.5;
Idouble = im2double(c);
avg = mean2(Idouble);
disp(avg)
sigma = std2(Idouble);
disp(sigma)
RGB2 = imadjust(Idouble,[avg-n*sigma,avg+n*sigma],[]);
figure();imshow(RGB2);
output='Final.jpg';
imwrite(RGB2, output);
 % final=c+A;
%figure();imshow(final);
```