Problem 1

myDisplayImageGrey.m

```
function myDisplayImageGrey(img,title1)
        imshow (mat2gray(img));
        title(title1);
    colormap(gray(200));
    axis on;
    colorbar;
```

myDisplayTwoImage.m

```
function myDisplayTwoImage(img1,img2)
        figure('Position', [100, 100, 1200, 600]);
        subplot(1,2,1);
        myDisplayImageGrey(img1,"Original Image");
        subplot(1,2,2);
        myDisplayImageGrey(img2,"Sharpened Image");
end
```

myLinearContrastStretching.m

```
function contrastImg = myLinearContrastStretching(input)
   c = size(input,3);
   contrastImg = zeros(size(input));
   input = double(input);
   for l=1:c
        maxIntensity = max(max(input(:,:,l)));
        minIntensity = min(min(input(:,:,l)));
        contrastImg(:,:,l) = (input(:,:,l) - minIntensity) * double(255) /
(maxIntensity - minIntensity);
```

```
  end
end
```

myMainScript.m

```
%% MyMainScript

tic;
%% Super Moon Crop
imageData = load('../data/superMoonCrop.mat');
image = imageData.imageOrig;
stretchedImage = myLinearContrastStretching(image);
sharpImage = myUnsharpMasking(image,1.1,9,1.1);
sharpImage = myLinearContrastStretching(sharpImage);
myDisplayTwoImage(stretchedImage,sharpImage);
%% Lion Crop
imageData = load('../data/lionCrop.mat');
image = imageData.imageOrig;
stretchedImage = myLinearContrastStretching(image);
sharpImage = myUnsharpMasking(image,1.1,9,1.6);
sharpImage = myLinearContrastStretching(sharpImage);
myDisplayTwoImage(stretchedImage,sharpImage);
toc;
```

myUnsharpMasking.m

```
function fiteredImage = myUnsharpMasking(input, sigma, windowSize,
scale)
        fiteredImage = input -
scale*imfilter(input,fspecial('log',windowSize,sigma));
end
```

Problem 2

myBilateralFiltering.m

```
function filteredImage = myBilateralFiltering(input, sigmaSpace,
sigmaIntensity,windowSize)

        % create pre defined filter gaussian space
        filter = fspecial('gaussian',2*windowSize+1 ,sigmaSpace);
        [len,wid] = size(input);
        filteredImage = zeros([len,wid]);
        for i = 1:len
        for j = 1:wid
        x = max(i-windowSize,1);
        X = min(i+windowSize,len);
        y = max(j-windowSize,1);
        Y = min(j+windowSize,wid);
        window = input(x:X,y:Y);
        x1 = max(2+windowSize-i,1);
        X1 = min(2*windowSize+1,len-i+windowSize+1);
        y1 = max(2+windowSize-j,1);
        Y1 = min(2*windowSize+1,wid-j+windowSize+1);
        intensity = exp(-((window-input(i,j)).^2)/(sigmaIntensity^2));
        singleTerm = filter((x1:X1),(y1:Y1)).*intensity;
        individualSum = window.*singleTerm;
        filteredImage(i,j) = sum(individualSum(:))/sum(singleTerm(:));
        end
        end
end
```

myDisplayImageGrey.m

```
function myDisplayImageGrey(img,title1)
        imshow (mat2gray(img));
        title(title1);
    colormap(gray(200));
    axis on;
    colorbar;
```

myDisplayThreeImage.m

```
function myDisplayThreeImage(img1,img2,img3)
        figure('Position', [100, 100, 1200, 600]);
        subplot(1,3,1);
        myDisplayImageGrey(img1,"Original Image");
        subplot(1,3,2);
        myDisplayImageGrey(img2,"Corrupted Image");
        subplot(1,3,3);
        myDisplayImageGrey(img3,"Filtered Image");
end
```

myMainScript.m

```
%% MyMainScript

tic;
%% Barbara
imageData = load('../data/barbara.mat');
image = imageData.imageOrig;
[len, wid] = size(image);
corruptedImage = image +
0.05*(max(max(image))-min(min(image)))*randn(len);
```

```
sigmaSpace = 1.6;
sigmaIntensity = 13.5;
windowSize = 4;
out = myBilateralFiltering(corruptedImage, sigmaSpace, sigmaIntensity,
windowSize);
rmsd = myRmsd(out,image);
display(rmsd);
myDisplayThreeImage(image,corruptedImage,out);

out1 = myBilateralFiltering(corruptedImage, 0.9*sigmaSpace,
sigmaIntensity, windowSize);
rmsd1 = myRmsd(out1,image);
display(rmsd1);

out2 = myBilateralFiltering(corruptedImage, 1.1*sigmaSpace,
sigmaIntensity, windowSize);
rmsd2 = myRmsd(out2,image);
display(rmsd2);

out3 = myBilateralFiltering(corruptedImage, sigmaSpace,
0.9*sigmaIntensity, windowSize);
rmsd3 = myRmsd(out3,image);
display(rmsd3);

out4 = myBilateralFiltering(corruptedImage, sigmaSpace,
1.1*sigmaIntensity, windowSize);
rmsd4 = myRmsd(out4,image);
display(rmsd4);

figure;
filter = fspecial('gaussian',2*windowSize+1 ,sigmaSpace);
imshow(filter,'InitialMagnification','fit');
```

```
title("Spatial Gaussian Mask")
colormap(gray(200));
axis on;
colorbar;

%% Grass
image = im2double(imread('../data/grass.png'));
[len, wid] = size(image);
corruptedImage = image +
0.05*(max(max(image))-min(min(image)))*randn(len);

sigmaSpace = 0.71;
sigmaIntensity = 0.31;
windowSize = 2;
out = myBilateralFiltering(corruptedImage, sigmaSpace, sigmaIntensity,
windowSize);
rmsd = myRmsd(out,image);
display(rmsd);
myDisplayThreeImage(image,corruptedImage,out);

out1 = myBilateralFiltering(corruptedImage, 0.9*sigmaSpace,
sigmaIntensity, windowSize);
rmsd1 = myRmsd(out1,image);
display(rmsd1);

out2 = myBilateralFiltering(corruptedImage, 1.1*sigmaSpace,
sigmaIntensity, windowSize);
rmsd2 = myRmsd(out2,image);
display(rmsd2);

out3 = myBilateralFiltering(corruptedImage, sigmaSpace,
0.9*sigmaIntensity, windowSize);
rmsd3 = myRmsd(out3,image);
```

```
display(rmsd3);

out4 = myBilateralFiltering(corruptedImage, sigmaSpace,
1.1*sigmaIntensity, windowSize);
rmsd4 = myRmsd(out4,image);
display(rmsd4);

figure;
filter = fspecial('gaussian',2*windowSize+1 ,sigmaSpace);
imshow(filter,'InitialMagnification','fit');
title("Spatial Gaussian Mask")
colormap(gray(200));
axis on;
colorbar;

%% Honey Comb Real
image = im2double(imread('../data/honeyCombReal.png'));
[len, wid] = size(image);
corruptedImage = image +
0.05*(max(max(image))-min(min(image)))*randn(len);

sigmaSpace = 0.82;
sigmaIntensity = 0.27;
windowSize = 3;
out = myBilateralFiltering(corruptedImage, sigmaSpace, sigmaIntensity,
windowSize);
rmsd = myRmsd(out,image);
display(rmsd);
myDisplayThreeImage(image,corruptedImage,out);

out1 = myBilateralFiltering(corruptedImage, 0.9*sigmaSpace,
sigmaIntensity, windowSize);
rmsd1 = myRmsd(out1,image);
```

```
display(rmsd1);

out2 = myBilateralFiltering(corruptedImage, 1.1*sigmaSpace,
sigmaIntensity, windowSize);
rmsd2 = myRmsd(out2,image);
display(rmsd2);

out3 = myBilateralFiltering(corruptedImage, sigmaSpace,
0.9*sigmaIntensity, windowSize);
rmsd3 = myRmsd(out3,image);
display(rmsd3);

out4 = myBilateralFiltering(corruptedImage, sigmaSpace,
1.1*sigmaIntensity, windowSize);
rmsd4 = myRmsd(out4,image);
display(rmsd4);

figure;
filter = fspecial('gaussian',2*windowSize+1 ,sigmaSpace);
imshow(filter,'InitialMagnification','fit');
title("Spatial Gaussian Mask")
colormap(gray(200));
axis on;
colorbar;

toc;
```

myRmsd.m

```
function rsmdVal = myRmsd(img1,img2)
        [len, wid] = size(img1);
        rsmdVal = sqrt(sum(sum((img1-img2).^2))/(len*wid));
end
```

Problem 3

myDisplayImageGrey.m

```
function myDisplayImageGrey(img,title1)
        imshow (mat2gray(img));
        title(title1);
    colormap(gray(200));
    axis on;
    colorbar;
```

myDisplayThreeImage.m

```
function myDisplayThreeImage(img1,img2,img3)
        figure('Position', [100, 100, 1200, 600]);
        subplot(1,3,1);
        myDisplayImageGrey(img1,"Original Image");
        subplot(1,3,2);
        myDisplayImageGrey(img2,"Corrupted Image");
        subplot(1,3,3);
        myDisplayImageGrey(img3,"Filtered Image");
end
```

myMainScript.m

```
%% MyMainScript

tic;

%% Filtering Barbara image
load('../data/barbara.mat');
imageOrig = imgaussfilt(imageOrig,0.66);
```

```
imageOrig = imresize(imageOrig,0.5);
[len, wid] = size(imageOrig);
imgCorrupt = imageOrig +
0.05*(max(max(imageOrig))-min(min(imageOrig)))*randn(len);

h = 1.26;
patchSize = [9,9];
windowSize = [25,25];
out = myPatchBasedFiltering(imgCorrupt, patchSize, windowSize, h);
rsmd0 = myRsmd(out,imageOrig);
display(rsmd0);
myDisplayThreeImage(imageOrig,imgCorrupt,out);

out1 = myPatchBasedFiltering(imgCorrupt, patchSize, windowSize, h*0.9);
rsmd1 = myRsmd(out1,imageOrig);
display(rsmd1);

out2 = myPatchBasedFiltering(imgCorrupt, patchSize, windowSize, h*1.1);
rsmd2 = myRsmd(out2,imageOrig);
display(rsmd2);

figure;
gaussianFilter = fspecial('gaussian', patchSize, double(patchSize(1))/6);
imshow(gaussianFilter,'InitialMagnification','fit');
title("Mask for Isotropic Patches")
colormap(gray(200));
axis on;
colorbar;


%% Filtering Grass image
imageOrig = im2double(imread('../data/grass.png'));
[len, wid] = size(imageOrig);
```

```
imgCorrupt = imageOrig +
0.05*(max(max(imageOrig))-min(min(imageOrig)))*randn(len);

h = 0.000109;
patchSize = [9,9];
windowSize = [25,25];
out = myPatchBasedFiltering(imgCorrupt, patchSize, windowSize, h);
rsmd0 = myRsmd(out,imageOrig);
display(rsmd0);
myDisplayThreeImage(imageOrig,imgCorrupt,out);

out1 = myPatchBasedFiltering(imgCorrupt, patchSize, windowSize, h*0.9);
rsmd1 = myRsmd(out1,imageOrig);
display(rsmd1);

out2 = myPatchBasedFiltering(imgCorrupt, patchSize, windowSize, h*1.1);
rsmd2 = myRsmd(out2,imageOrig);
display(rsmd2);

figure;
gaussianFilter = fspecial('gaussian', patchSize, double(patchSize(1))/6);
imshow(gaussianFilter,'InitialMagnification','fit');
title("Mask for Isotropic Patches")
colormap(gray(200));
axis on;
colorbar;

%% Filtering Honey Comb image
imageOrig = im2double(imread('../data/honeyCombReal.png'));
[len, wid] = size(imageOrig);
imgCorrupt = imageOrig +
0.05*(max(max(imageOrig))-min(min(imageOrig)))*randn(len);
```

```
h = 0.00013;
patchSize = [9,9];
windowSize = [25,25];
out = myPatchBasedFiltering(imgCorrupt, patchSize, windowSize, h);
rsmd0 = myRsmd(out,imageOrig);
display(rsmd0);
myDisplayThreeImage(imageOrig,imgCorrupt,out);

out1 = myPatchBasedFiltering(imgCorrupt, patchSize, windowSize, h*0.9);
rsmd1 = myRsmd(out1,imageOrig);
display(rsmd1);

out2 = myPatchBasedFiltering(imgCorrupt, patchSize, windowSize, h*1.1);
rsmd2 = myRsmd(out2,imageOrig);
display(rsmd2);

figure;
gaussianFilter = fspecial('gaussian', patchSize, double(patchSize(1))/6);
imshow(gaussianFilter,'InitialMagnification','fit');
title("Mask for Isotropic Patches")
colormap(gray(200));
axis on;
colorbar;

toc;
```

myPatchBasedFiltering.m

```
function filterImg = myPatchBasedFiltering(img, patchSize, windowSize, h)
    [len, wid] = size(img);

    paddedImage = padarray(img, [(patchSize(1)-1)/2, (patchSize(2)-1)/2],
'symmetric', 'both');
```

```
    gaussianFilter = fspecial('gaussian', patchSize, double(patchSize(1))/6);
    gaussianFilter = gaussianFilter/(sum(gaussianFilter(:)));
    gaussianFilterVector = gaussianFilter(:);

    featureImage = im2col(paddedImage, patchSize);
    featureImage = bsxfun(@times, featureImage, gaussianFilterVector);
    % Feature Image is of the same size as of the original image with last
dimension equal to feature dimension
    featureImage = permute(reshape(featureImage,
[patchSize(1)*patchSize(2), len, wid]), [2,3,1]);

    constant1 = (windowSize(1)-1)/2;
    constant2 = (windowSize(1)+1)/2;

    filterImg = zeros(size(img));

    for i=1:len
        for j=1:wid
            minI = (i-constant1);
            maxI = (i+constant1);
            minJ = (j-constant1);
            maxJ = (j+constant1);
            if i < constant2
                minI = 1;
            elseif i > (len-constant1)
                maxI = len;
            end
            if j < constant2
                minJ = 1;
            elseif j > (wid-constant1)
                maxJ = wid;
            end
```

```
                patch = featureImage(minI:maxI, minJ:maxJ, :);
                sumSquareDiffernce = sum(bsxfun(@minus, patch,
featureImage(i,j,:)).^2, 3);
                expSumSquareDiffernce = exp(sumSquareDiffernce*(-1.0/h));
                dotWiseMult = img(minI:maxI,
minJ:maxJ).*expSumSquareDiffernce;
                filterImg(i,j) =
sum(dotWiseMult(:))/sum(expSumSquareDiffernce(:));
        end
   end
end
```

myRsmd.m

```
function rsmdVal = myRsmd(img1,img2)
        [len, wid] = size(img1);
        rsmdVal = sqrt(sum(sum((img1-img2).^2))/(len*wid));
end
```