# CS-663 Assignment 2 Q3

Soham Naha (193079003)
Akshay Bajpai (193079002)
Mohit Agarwala (19307R004)

September 29, 2020

## 3 (50 points) Edge-preserving Smoothing using Patch-Based Filtering.

Input image:

1. *3/data/barabara.mat* .
2. *3/data/grass.png* .
3. *3/data/honeyCombReal.png* .

Redo the previous problem using patch-based filtering. If you think your code takes too long to run, (i) resize the image by subsampling by a factor of 2 along each dimension, after applying a Gaussian blur of standard deviation around 0.66 pixel width and (ii) apply the filter to the resized image.
Use 9 9 patches. Use a Gaussian, or clipped Gaussian, weight function on the patches to make the patch more isotropic (as compared to a square patch). Note: this will imply neighbor-location-weighted distances between patches. For filtering pixel "p", use patches centered at pixels "q" that lie within a window of size approximately 25 25 around "p".

- Write a function *myPatchBasedFiltering.m* to implement this.
- Show the original, corrupted, and filtered versions side by side, using the same (gray) colormap.
- Show the mask used to make patches isotropic, as an image.
- Report the optimal parameter value found, say , along with the optimal RMSD.
- Report RMSD values for filtered images obtained with (i) 0.9 and (ii) 1.1 , with all other parameter values unchanged.

**Solution:**

**Note:** Only the third question of the assignment has been done in Matlab in contrary to the first two questions which are done in Python. We tried to implement the same algorithm in python , but it turned out that the run time was very high and it was difficult to go through several iterations to find the optimal set of values. So the same algorithm was then coded in Matlab and the runtime went down considerably. This has been done post permission from Prof.S Awate.

**Main Script:**

```
tic;
 sigma_star=1.5;

 for i=[sigma_star , 1.1*sigma_star ,0.9*sigma_star]
 image = load('../data/grassNoisy.mat');
 imageArray = image.imgCorrupt;
 filtered= myPatchBasedFiltering(imageArray.*255,9,25,4,i);
 original = imread('../data/grass.png');
 original_double = im2double(original);
 h=figure();
```

```
 subplot(1,3,1);
 imshow(imageArray,[]);
 title('Noisy Image');
 subplot(1,3,2);
 imshow(original_double,[]);
 title('Original Image');
 subplot(1,3,3);
 imshow(filtered,[]);
 title('Filtered Image');
 file_name = strcat('../Images/Grass_filtered_',num2str(i));
 saveas(h, file_name, 'jpg');
 rmsd_calc(original_double.*255,filtered);
 end

 sigma_star=1.25;
 for i=[sigma_star ,1.1*sigma_star,0.9*sigma_star]
 image = load('../data/honeyCombReal_Noisy.mat');
 imageArray = image.imgCorrupt;
 filtered=myPatchBasedFiltering(imageArray.*255,9,25,5,i);
 original = imread('../data/honeyCombReal.png');
 original_double = im2double(original);
 h=figure();
 subplot(1,3,1);
 imshow(imageArray,[]);
 title('Noisy Image');
 subplot(1,3,2);
 imshow(original_double,[]);
 title('Original Image');
 subplot(1,3,3);
 imshow(filtered,[]);
 title('Filtered Image');
 file_name = strcat('../Images/Honey_filtered_',num2str(i));
 saveas(h,file_name , 'jpg');
 rmsd_calc(original_double.*255,filtered);
 end

image = load('../data/barbara.mat');
image_orig = image.imageOrig;
img = image_orig/(max(max(image_orig)));

rng(1);
noise = randn(size(img, 1)) * 0.05 * max(max(img));
imgn  = max(0, img+noise);
sigma_star = 1.5;
 for i= [sigma_star,1.1*sigma_star,0.9*sigma_star]

     filtered= myPatchBasedFiltering(imgn.*255,9,25,5,i);
     original = img;
     h=figure();
     subplot(1,3,1);
     imshow(img,[]);
     title('Original Image');
     subplot(1,3,2);
     imshow(imgn,[]);
     title('Noisy Image');
     subplot(1,3,3);
     imshow(filtered,[]);
     title('Filtered Image');
     file_name = strcat('../Images/Barbara_filtered_',num2str(i));
     saveas(h, file_name, 'jpg');
     rmsd_calc(original,filtered);
 end

patch_isotropy = fspecial('gaussian', [9,9], 4);
h=figure();
```

```
imshow(patch_isotropy,[]);
saveas(h,'../Images/isotropy_patch_sd4','jpg');
patch_isotropy = fspecial('gaussian', [9,9], 5);
figure();
imshow(patch_isotropy,[]);
saveas(h,'../Images/isotropy_patch_sd5','jpg');
toc;
```

```
function filtered_image=myPatchBasedFiltering(imageArray,patch,window_size,sigma_space,h)

    [r,c]=size(imageArray);
    box_filter = zeros(patch,patch);
    mid_point = round(window_size/2);

    padded_image = zeros(r+window_size-1,c+window_size-1);
    padded_image(mid_point:r+mid_point-1,mid_point:c+mid_point-1)= imageArray(:,:);
    filtered_image = zeros(r+window_size-1,c+window_size-1);


    patch_mid = round(patch/2);

    for i=mid_point:r+mid_point-1
        for j=mid_point:c+mid_point-1

            image_part = 1.*padded_image(i-patch_mid+1:i+patch_mid-1,j-patch_mid+1 : j+patch_mid-1);
            window_image_part = padded_image(i-mid_point+1:i+mid_point-1,j-mid_point+1 : j+mid_point-1);
            filtered_image(i,j) = mean_filter(image_part,window_image_part,patch,sigma_space,h);


        end
    end

    filtered_image = filtered_image(mid_point:r+mid_point-1,mid_point:c+mid_point-1);
```

```
function filtered_image=mean_filter(image_patch,imageArray,window_size,sigma_space,h)

    [r,c]=size(imageArray);

    mid_point = round(window_size/2);

    padded_image = zeros(r+window_size-1,c+window_size-1);
    padded_image(mid_point:r+mid_point-1,mid_point:c+mid_point-1)= imageArray(:,:);

    patch_isotropy = fspecial('gaussian', [window_size, window_size], sigma_space);
    sum_=0;
    den_=0;
    for i=mid_point:r+mid_point-1
        for j=mid_point:c+mid_point-1

            image_part = padded_image(i-mid_point+1:i+mid_point-1,j-mid_point+1 : j+mid_point-1);
            diff = exp(-1 * sum(sum((patch_isotropy.*(image_patch -image_part)).^2)) / (h^2));
            sum_ =  sum_+padded_image(i,j)*diff;
            den_ =den_+diff;
        end
    end

    filtered_image = sum_/den_;
```

```
function rmsd_calc(img1,img2)
img1 = img1 / max(max(img1));
img2 = img2 / max(max(img2));
```

```
4
5  diff = img1 - img2;
6  rmsd = sqrt(sum(sum(diff.^2)) / numel(diff));
7  display(rmsd);
```

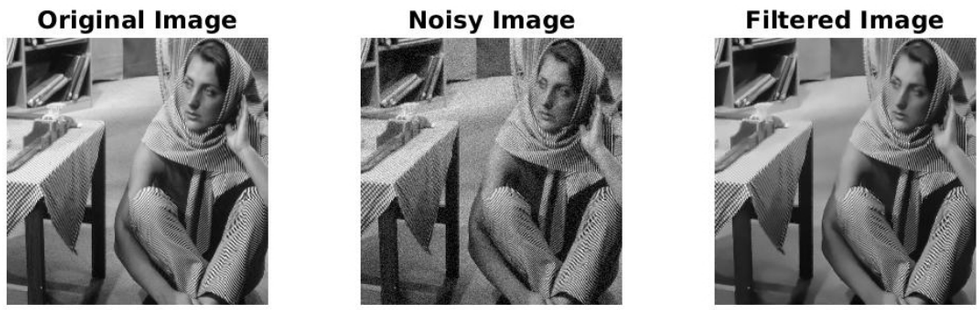


Figure 1: Output of Patch based filtering on Barbara image

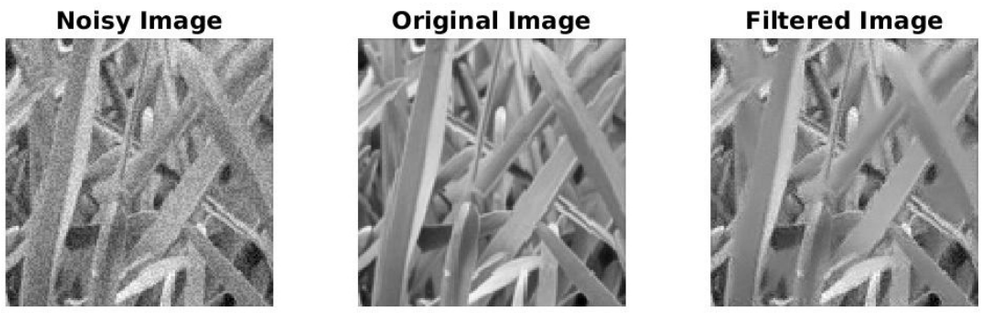


Figure 2: Output of Patch based filtering on Grass image

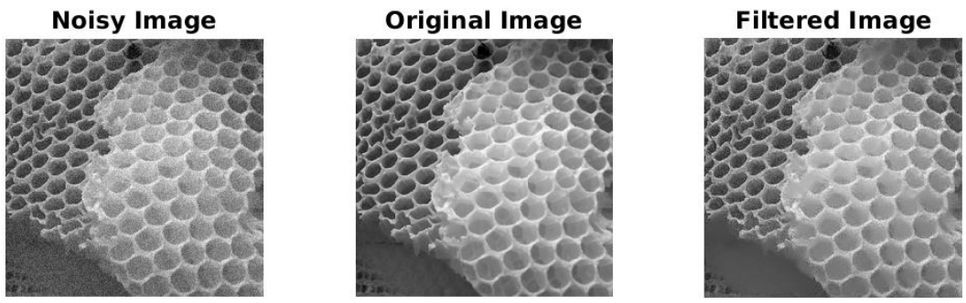


Figure 3: Output of Patch based filtering on Honeycomb image

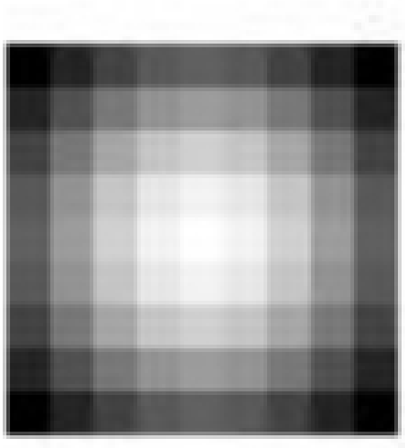

Figure 4: (a) Isotropy mask used for all the images $\sigma^*_{space}$=4

**RMSD Calculations :**

- 3/data/barbara.mat
  The optimum values of parameters found are :
  - $\sigma^*_{space} = 1.5$
  - RMSD of barbara image for $\sigma^*_{space}$= 1.5 is 0.0476.

  RMSD calculations for the other values specified:
  1. RMSD of barbara image for Sigma Spatial(0.9*$\sigma^*_{space}$): 1.35 is 0.0488.
  2. RMSD of barbara image for Sigma spatial(1.1*$\sigma^*_{space}$): 1.65 is 0.0499.

- 3/data/grass.png
  The optimum values of parameters found are :
  - $\sigma^*_{space} = 1.5$
  - RMSD of barbara image for $\sigma^*_{space}$= 1.5 is 0.0395.

  RMSD calculations for the other values specified:
  1. RMSD of barbara image for Sigma Spatial(0.9*$\sigma^*_{space}$): 1.35 is 0.0460.
  2. RMSD of barbara image for Sigma spatial(1.1*$\sigma^*_{space}$): 1.65 is 0.0420.

- 3/data/honeyCombReal.png
  The optimum values of parameters found are :
  - $\sigma^*_{space} = 1.25$
  - RMSD of barbara image for $\sigma^*_{space}$= 1.25 is 0.0449.

  RMSD calculations for the other values specified:
  1. RMSD of barbara image for Sigma Spatial(0.9*$\sigma^*_{space}$): 1.125 is 0.0467.
  2. RMSD of barbara image for Sigma spatial(1.1*$\sigma^*_{space}$): 1.375 is 0.0452.

**Note** :All the images corresponding to the optimal value of $\sigma$ and other values are present in the images folder.