CS663: Digital Image Processing
Assignment 02: Filtering

Manas Vashistha, 17D070064
Mohd Safwan, 17D070047
Sumrit Gupta, 170040044

September 2, 2019

1 Problem Statement 2

(30 points) Edge-preserving Smoothing using Bilateral Filtering.

Input images:

1. 2/data/barbara.mat
2. 2/data/grass.png
3. 2/data/honeyCombReal.png

- Write a function myBilateralFiltering.m to implement this.
- Show the original, corrupted, and filtered versions side by side, using the same (gray) colormap.
- Show the mask for the spatial Gaussian, as an image.
- Report the optimal parameter values found, say σ^*_{space} and $\sigma^*_{intensity}$, along with the optimal space intensity RMSD.
- Report RMSD values for filtered images obtained with (i) $0.9\sigma^*_{space}$ and $\sigma^*_{intensity}$, (ii) $1.1\sigma^*_{space}$ and $\sigma^*_{intensity}$, (iii) σ^*_{space} and $0.9\sigma^*_{intensity}$, and (iv) σ^*_{space} and 1.1 $\sigma^*_{intensity}$,with all other parameter and space intensity values unchanged.

1.1 myBilateralFiltering.m

```
function [corrupted_img, mask, output_image, rmsd, sig]  = myBilateralFiltering(img, sig_s, sig_r, window)

[r, c] = size(img);
% max_int = max(max(img));
img_scaled = img;
% ./max_int;
sig = double(max(max(img_scaled)) - min(min(img_scaled)))/20;
mask = sig * randn(r, r, 1);
corrupted_img = img_scaled + mask;
output_image = zeros(r, c);
w = round((window-1)/2);
corrupted_img1 = padarray(corrupted_img, [w, w], 'replicate');
h = waitbar(0,'Applying bilateral filter...');
set(h,'Name','Bilateral Filter Progress');
for i = w+1:w+r
    for j = w+1:w+c
        k = i-w:i+w;
        l = j-w:j+w;
        f_r = (exp(-((corrupted_img1(k, l) - corrupted_img1(i, j)).^2)./(2*(sig_r)^2)))./sqrt(2*pi*sig_r^2);
        [k_r, l_c] = find(corrupted_img1(k, l));
        g_s = reshape(exp(-((k_r).^2 + (l_c).^2)./(2*(sig_s)^2))./sqrt(2*pi*sig_s^2), size(f_r));
        w_p = sum(f_r .*g_s, 'all');
        I_i = corrupted_img1(k, l) .* f_r .* g_s;
        output_image(i-(w), j-(w)) = (sum(I_i, 'all')/w_p);
    end
waitbar(i/r);
end
close(h);

rmsd = sqrt(sum((img_scaled - output_image).^2, 'all')/(r*c));
disp(rmsd);

end
```

Figure 1: myBilateralFiltering.m

$$Z_i \sim \mathcal{N}(0, N) \tag{1}$$

$$Y_i = X_i + Z_i \tag{2}$$

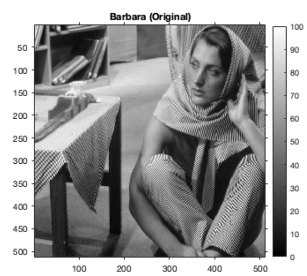
$$I^{filtered}(x) = \frac{1}{W_p} \sum_{x_i \epsilon \Omega} I(x_i) f_r(||I(x_i) - I(x)||) g_s(||x_i - x||) \tag{3}$$

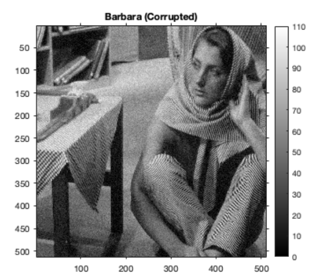
$$W_p = \sum_{x_i \epsilon \Omega} f_r(||I(x_i) - I(x)||) g_s(||x_i - x||) \tag{4}$$

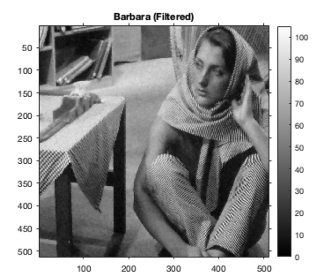
$$RMSD = \sqrt{\frac{1}{N} \sum_{p \epsilon I} (A(p)^2 - B(p)^2)} \tag{5}$$

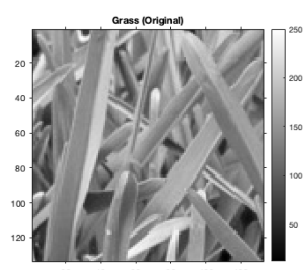
The code `myBilateralFiltering.m` first introduces a White Gaussian Noise(Z_i) in the uncorrupted image(X_i) to get the corrupted image(Y_i) and then implements the above two equations (3) and, (4) on the input corrupted image to generate an uncorrupted, filtered image.
The dissimalarity in the the initial uncorrupted input image and the finally filtered output image is calculated using a metric called *Root Mean Squared Difference* (RMSD). The optimization process requires decresing the value of RMSD.

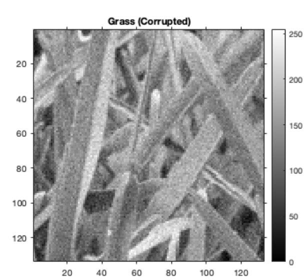
1.2 Images & Results

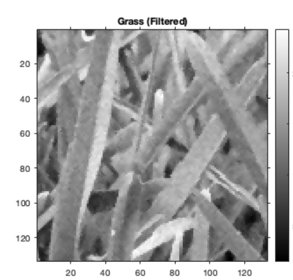




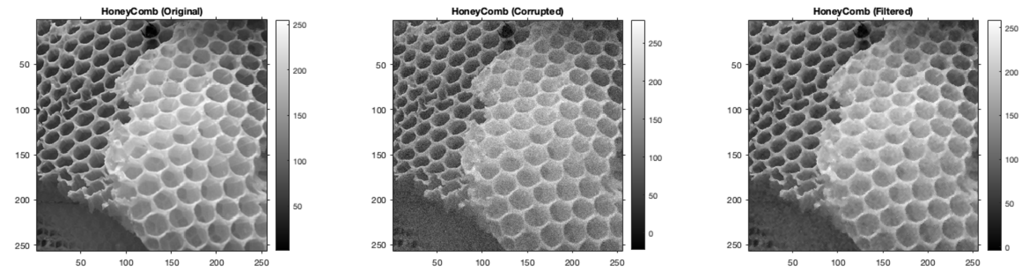








The above three figures display the uncorrupted input image, the corrupted image and, the finally filtered output image side by side for the input images: 'Barbara.mat', 'Grass.png' and, 'HoneyCombReal.png' respectively.



1.3 Masks of spatial Gaussian

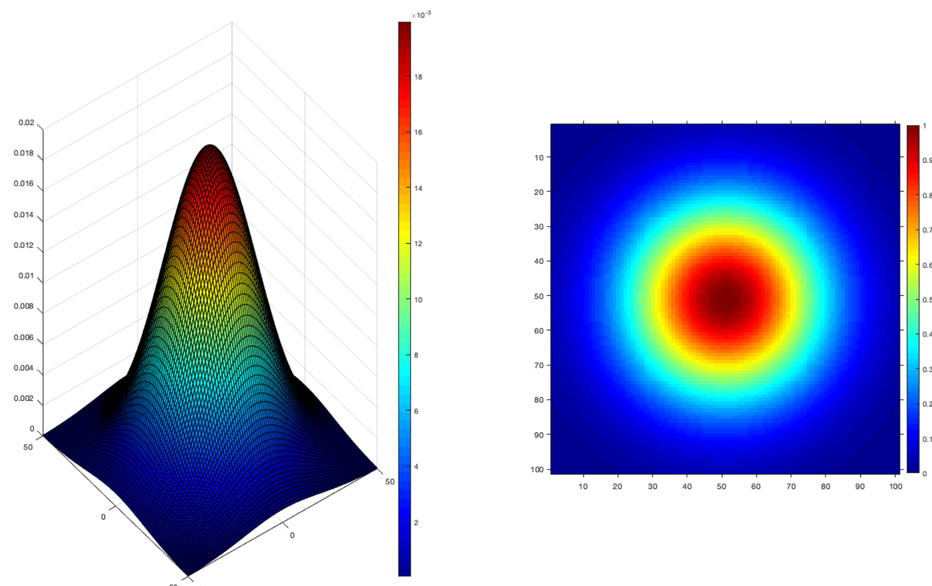


Figure 2: Mask of spatial sigma Gaussian for Barbara

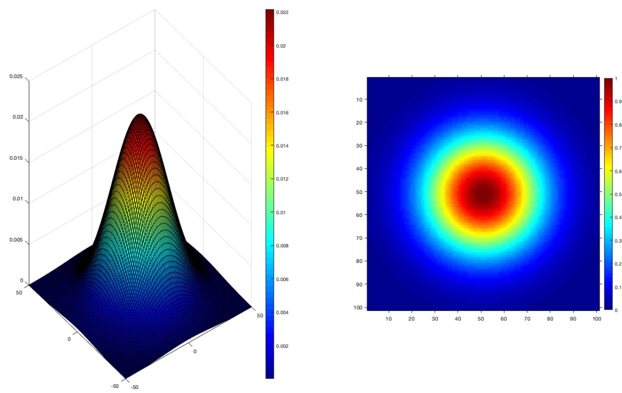


Figure 3: Mask of spatial sigma Gaussian for grass

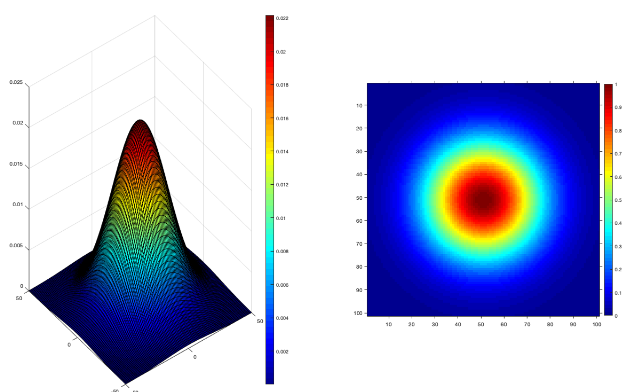


Figure 4: Mask of spatial sigma Gaussian for honeyCombReal

The above images are the plots of Zero Mean 2d Gaussians with σ value being equal to the spatial sigmas of the respective images.

1.4 Results

The given table shows the values of σ^*_{space} & $\sigma^*_{intensity}$ of each image for which the RMSD of the image attains its minimum value.

Table 1: Optimal parameter values for the test images

Image	σ^*_{space}	$\sigma^*_{intensity}$	Window Size	RMSD
Barbara	20	9	5	3.3340
Grass	20	27.5	3	7.3751
HoneyCombReal	20	33.275	3	7.2710

1.5 Experiments

Tables below show the variation in RMSD for the input images as the $\sigma^*_{intensity}$ & σ^*_{space} are varied individually keeping all the other parameter values constant.

1.5.1 $\sigma^*_{intensity}$ & $\mathit{0.9}\sigma^*_{space}$

Table 2: $\sigma^*_{intensity}$ & $\mathit{0.9}\sigma^*_{space}$

Image	$\mathit{0.9}\sigma^*_{space}$	$\sigma^*_{intensity}$	Window Size	RMSD
Barbara	18	9	5	3.3448
Grass	16.2	25	3	7.5060
HoneyCombReal	16.2	25	3	7.2885

1.5.2 $\sigma^*_{intensity}$ & $\mathit{1.1}\sigma^*_{space}$

Table 3: $\sigma^*_{intensity}$ & $\mathit{1.1}\sigma^*_{space}$

Image	$\mathit{1.1}\sigma^*_{space}$	$\sigma^*_{intensity}$	Window Size	RMSD
Barbara	22	9	5	3.3645
Grass	19.8	25	3	7.5988
HoneyCombReal	19.8	25	3	7.3828

1.5.3 ***0.9***$\sigma^*_{intensity}$ & σ^*_{space}

Table 4: *0.9*$\sigma^*_{intensity}$ & σ^*_{space}

Image	σ^*_{space}	*0.9*$\sigma^*_{intensity}$	Window Size	RMSD
Barbara	20	8.1	5	3.3598
Grass	18	24.75	3	7.5668
HoneyCombReal	18	29.95	3	7.4096

1.5.4 ***1.1***$\sigma^*_{intensity}$ & σ^*_{space}

Table 5: *1.1*$\sigma^*_{intensity}$ & σ^*_{space}

Image	σ^*_{space}	*1.1*$\sigma^*_{intensity}$	Window Size	RMSD
Barbara	20	9.9	5	3.3551
Grass	18	30.25	3	7.4235
HoneyCombReal	18	36.6	3	7.2954