DHRUV GARG 16 BCE 1190

IMAGE NOISE MODELS AND RESTORATION USING FILTERS

DIGITAL ASSIGNMENT 2

QUESTION 1

NOTE: For the sake of uniformity and simplicity, all the images have been taken as grayscale images.

GAUSSIAN NOISE AND ITS RESOTRATION

```
%Reading original image
original_image = imread('sundar_pichai.jpg');
%Converting image to grayscale
GrayImg = uint8(0.2989 * original_image(:,:,1) + 0.5870 * original_image(:,:,2) +
0.1140 * original_image(:,:,3));
imshow(GrayImg);
title('Original image'); figure;

%Applying Gaussian noise
gaussian_noise = imnoise(GrayImg, 'gaussian', 0, 0.01);
imshow(gaussian_noise);
title('Gaussian noise mean=0, variance=0.1');

%Applying wiener filter to remove Gaussian noise
wiener_filtered = wiener2(gaussian_noise, [10 10]);
figure;
imshow(wiener_filtered);
title('Gaussian noise removed by Wiener filter');

%Applying median filter to remove Gaussian noise
Kmedian_filtered = medfilt2(gaussian_noise, [10 10]);
figure;
imshow(Kmedian_filtered);
title('Gaussian noise removed by Median Filter');
```

Original image

Gaussian noise mean=0, variance=0.1

Gaussian noise removed by Wiener filter

Gaussian noise removed by Median Filter

Adding Gaussian noise to image

I have used the pre-defined function **imnoise()** and given parameters as ('gaussian', 0 and 0.01) where mean is 0 and variance is 0.01.

Justification for wiener filter and median filter used

The Wiener filter (a type of linear filter) tailors itself to the local image variance. Where the variance is large, wiener2 performs little smoothing. Where the variance is small, wiener2 performs more smoothing. The median filter causes some amount of blurring while preserving the edges. It does a fairly good job in removing the Gaussian noise. In this case, we see that Wiener filter performs better. I have used the pre-defined **wiener2()** and **medfilt2().**

SALT-PEPPER NOISE AND ITS RESTORATION

```
original_image = imread('sundar_pichai.jpg');
%Converting image to grayscale
GrayImg = uint8(0.2989 * original_image(:,:,1) + 0.5870 * original_image(:,:,2) +
0.1140 * original_image(:,:,3));
imshow(GrayImg); title('Original image'); figure;

%Applying Salt and Pepper noise
saltpepper_noise = imnoise(GrayImg, 'salt & pepper', 0.06);
imshow(saltpepper_noise);
title('Salt and Pepper noise image'); figure;

%Applying Average filter to remove salt and pepper noise
Kaverage_filtered = filter2(fspecial('average',3),saltpepper_noise)/255;
imshow(Kaverage_filtered);
title('Salt and Pepper noise removed by Average Filter'); figure;

%Applying median filter to remove Salt and Pepper noise
Kmedian_filtered = medfilt2(saltpepper_noise);
imshow(Kmedian_filtered);
title('Salt and Pepper noise removed by Median Filter');
```

Salt and Pepper noise image

Salt and Pepper noise removed by Average Filter

Adding Salt-Pepper noise to image

I used the pre-defined function **imnoise()** with parameters ('salt & pepper', 0.06) to apply additive Salt and Pepper noise with the value of noise density(d) as 0.06.

Justification for average filter and median filter used

These two types of filtering set the value of the output pixel to the average of the pixel values in the neighbourhood around the corresponding input pixel. However, with median filtering, the value of an output pixel is determined by the median of the neighbourhood pixels, rather than the mean. The median is much less sensitive than the mean to extreme values (called outliers). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image. Used the pre-defined functions **filter2()** and **medfilt2().**

RAYLEIGH NOISE AND ITS RESTORATION

```
%Reading original image
original_image = imread('sundar_pichai.jpg');
%Converting image to grayscale
GrayImg = uint8(0.2989 * original_image(:,:,1) + 0.5870 * original_image(:,:,2) +
0.1140 * original_image(:,:,3));
GrayImg = im2double(GrayImg);
imshow(GrayImg);
title('Original image'); figure;

%Adding rayleigh noise
noise = raylrnd(0.2,720,1280);
%imshow(noise);
%title('Noise to be added');
%figure;
rayleigh_Noise = GrayImg + noise;
imshow(rayleigh_Noise);
title('Rayleigh noise image'); figure;
%Applying median filter to remove rayleigh noise
```

```
median_filter = ordfilt2(rayleigh_Noise, 5, ones(3,3));
imshow(median_filter);
title('Rayleigh noise removed by median filter'); figure;

%Applying minimum filter to remove rayleigh noise
minimum_filter = ordfilt2(rayleigh_Noise,1,ones(3,3));
imshow(minimum_filter);
title('Rayleigh noise removed by minimum filter');
```

Rayleigh noise removed by median filter

Rayleigh noise removed by minimum filter

Adding rayleigh noise to image

Used function **raylrnd()** with parameters (0.2, 720,1280) where 0.2=parameter value and 720x1280 the dimension.

Justification for median filter and minimum filter used

While median filter provided good results, minimum filter performs even better. Minimum filter is a morphological filter, which replaces the central pixel with the darkest one in the running window. I have used the pre-defined function **ordfilt2()** to perform 2D order statistic filtering.

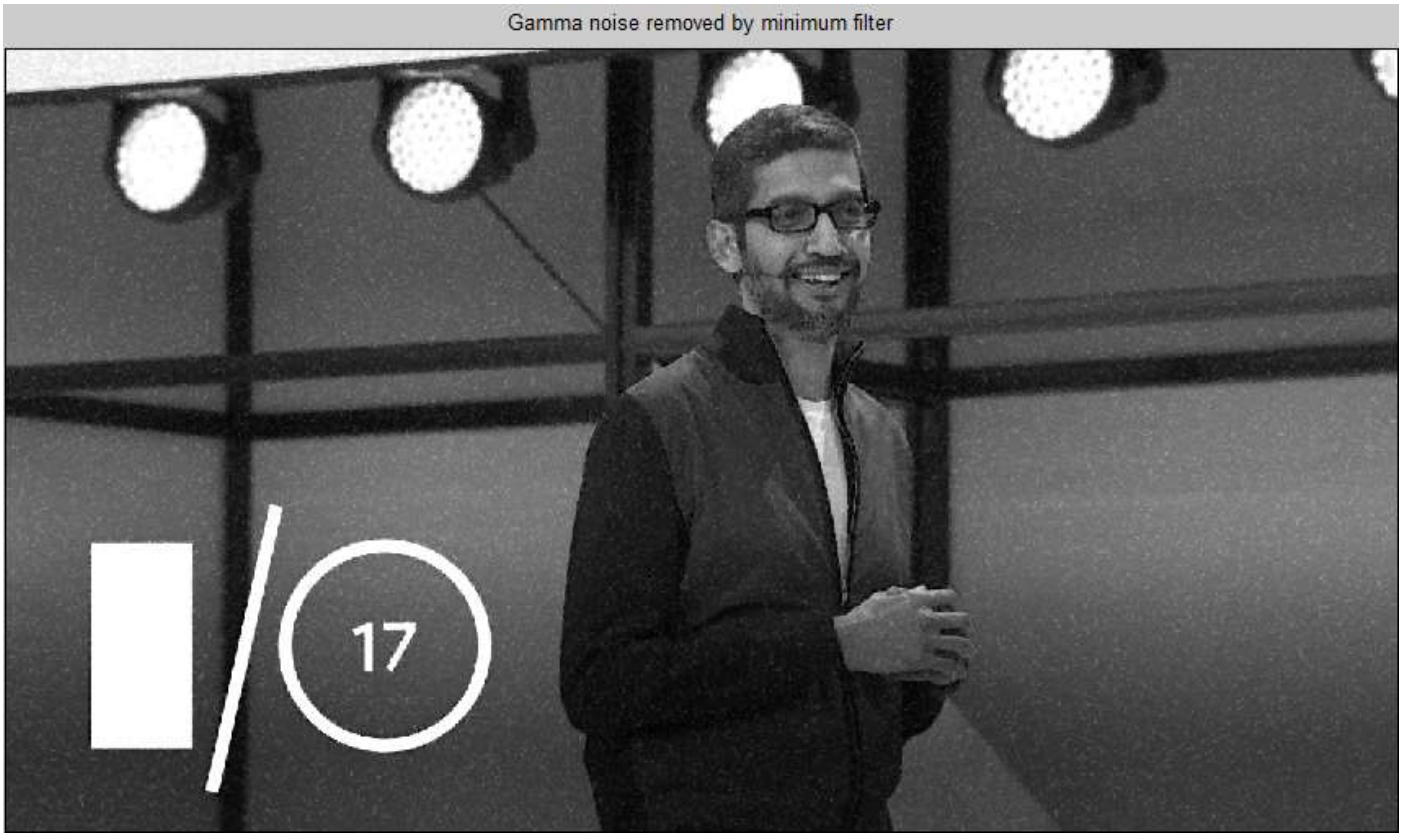
DHRUV GARG 16 BCE 1190

GAMMA NOISE AND ITS RESTORATION

```
%Reading original image
original_image = imread('sundar_pichai.jpg');
%Converting image to grayscale
GrayImg = uint8(0.2989 * original_image(:,:,1) + 0.5870 * original_image(:,:,2) +
0.1140 * original_image(:,:,3));
GrayImg = im2double(GrayImg);
imshow(GrayImg);
title('Original image');
figure;

%Applying gamma noise
a = 5; b = 1;
k = -1/a;
noise = zeros(720,1280);
for j = 1:b
    noise = noise + k*log(1 - rand(720,1280));
end
%imshow(noise);
%title('Noise');
%figure;
gamma_noise = GrayImg + noise;
imshow(gamma_noise);
title('Gamma noise image');
figure;

%Applying median filter to remove gamma noise
median_filter = ordfilt2(gamma_noise, 5, ones(3,3));
imshow(median_filter);
title('Gamma noise removed by median filter');
figure;

%Applying minimum filter to remove gamma noise
minimum_filter = ordfilt2(gamma_noise,1,ones(3,3));
imshow(minimum_filter);
title('Gamma noise removed by minimum filter');
```

Gamma noise image

Gamma noise removed by median filter



Adding gamma noise to image

I have used the formula **noise = noise + k*log(1 - rand(720,1280))** (this statement is repeated inside a loop) which forms the probability distribution for the noise. This noise is then added to the gray-scale image. Parameter to the rand(720,1280) denotes the dimension of the image.

Justification for median filter and minimum filter used

While median filter provided good results, minimum filter performs even better for Gamma noise. Minimum filter is a morphological filter, which replaces the central pixel with the darkest one in the running window. I have used the pre-defined function **ordfilt2()** to perform 2D order statistic filtering.

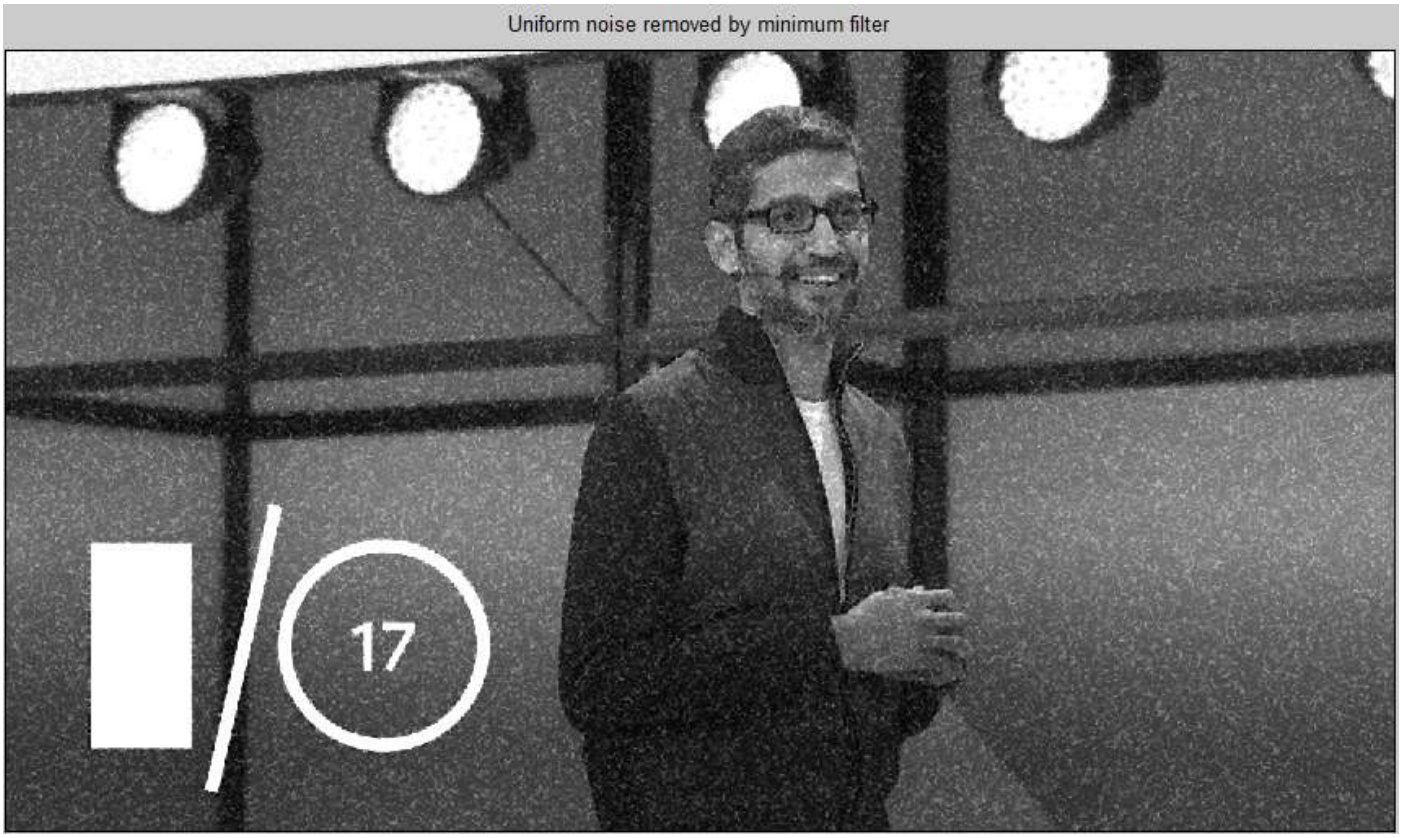
UNIFROM NOISE AND ITS RESTORATION

```
%Reading original image
original_image = imread('sundar_pichai.jpg');
%Converting image to grayscale
GrayImg = uint8(0.2989 * original_image(:,:,1) + 0.5870 * original_image(:,:,2) +
0.1140 * original_image(:,:,3));
GrayImg = im2double(GrayImg);
imshow(GrayImg);
title('Original image');
figure;

%Applying uniform noise
a = 0; b = 0.6;
noise = a + (b-a)*rand(720,1280);
%imshow(noise);
%title('Noise');
%figure;
uniform_noise = GrayImg + noise;
imshow(uniform_noise);
title('Uniform noise image'); figure;
```

```
%Applying median filter to remove uniform noise
median_filter = ordfilt2(uniform_noise, 5, ones(3,3));
imshow(median_filter);
title('Uniform noise removed by median filter'); figure;

%Applying minimum filter to remove uniform noise
minimum_filter = ordfilt2(uniform_noise,1,ones(3,3));
imshow(minimum_filter);
title('Uniform noise removed by minimum filter');
```

Uniform noise removed by maximum filter

Uniform noise removed by median filter



Adding uniform noise to image

I have used the formula **noise = a + (b-a)*rand(720,1280)** which forms the probability distribution for the noise. This noise is then added to the gray-scale image. Parameter to the rand(720,1280) denotes the dimension of the image.

Justification for maximum, median and minimum filter used

While maximum filter wasn't of much use and distorted the image further, the median filter provided good results, and minimum filter performed the best. Minimum filter is a morphological filter, which replaces the central pixel with the darkest one in the running window. I have used the pre-defined function **ordfilt2()** to perform 2D order statistic filtering.

EXPONENTAIL NOISE AND ITS RESTORATION

```
%Reading original image
original_image = imread('sundar_pichai.jpg');
%Convering image to grayscale
GrayImg = uint8(0.2989 * original_image(:,:,1) + 0.5870 * original_image(:,:,2) +
0.1140 * original_image(:,:,3));
GrayImg = im2double(GrayImg);
imshow(GrayImg);
title('Original image');
figure;

%Applying exponential noise
a = 8;
k = -1/a;
noise = k*log(1 - rand(720,1280));
%imshow(noise);
%title('Noise');
%figure;
exp_noise = GrayImg + noise;
imshow(exp_noise);
```

```
title('Exponential noise image');
figure;

%Applying maximum filter to remove exponential noise
maximum_filter = ordfilt2(exp_noise, 9, ones(3,3));
imshow(maximum_filter);
title('Exponential noise removed by maximum filter');
figure;

%Applying median filter to remove exponential noise
median_filter = ordfilt2(exp_noise, 5, ones(3,3));
imshow(median_filter);
title('Exponential noise removed by median filter');
figure;

%Applying minimum filter to remove exponential noise
minimum_filter = ordfilt2(exp_noise,1,ones(3,3));
imshow(minimum_filter);
title('Exponential noise removed by minimum filter');
```

Exponential noise image

Exponential noise removed by median filter

Adding exponential noise to image

I have used the formula **noise = k*log(1 - rand(720,1280))** which forms the probability distribution for the noise. This noise is then added to the gray-scale image. Parameter to the rand(720,1280) denotes the dimension of the image.

Justification for median and minimum filter used

The median filter provided good results, and minimum filter performed the best. Minimum filter is a morphological filter, which replaces the central pixel with the darkest one in the running window. I have used the pre-defined function **ordfilt2()** to perform 2D order statistic filtering.

QUESTION 2

Meaning of SNR value

The signal-to-noise ratio (SNR) is used in imaging as a physical measure of the sensitivity of a imaging system. SNR is measured in decibels (dB) of power. An image with SNR:32.04 dB = excellent image quality and SNR:20 dB = acceptable image quality. Thus an **image with low SNR is not as acceptable as an image with high SNR.**

Peak signal-to-noise ratio, often abbreviated PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation.

Method for calculation of SNR and peak SNR value

To find the SNR values from different filters, I wanted to use the **pre-defined psnr() function**, which required the **filtered image and original/noiseless image** to be **passed as parameters.** Hence, I took a coloured image, converted it to grayscale. This is the original image. This noiseless image, when passed through different filters, gave the required snr and peak-snr values.

Code to create noisy image – used uniform noise

```
%Reading original image
original_image = imread('Rio_de_janeiro.jpg');
%Converting image to grayscale
GrayImg = uint8(0.2989 * original_image(:,:,1) + 0.5870 * original_image(:,:,2) +
0.1140 * original_image(:,:,3));
GrayImg = im2double(GrayImg);
imshow(GrayImg);
title('Original image');
figure;

imwrite(GrayImg,'RioDeJaneiro.jpg','jpg');

%Applying uniform noise
a = 0; b = 0.5;
noise = a + (b-a)*rand(1000,1600);
%imshow(noise);
%title('Noise');
%figure;
uniform_noise = GrayImg + noise;
imshow(uniform_noise);
title('Uniform noise image');
figure;

imwrite(uniform_noise,'RioDeJaneiro_noise.jpg','jpg');
```

Original image

Uniform noise image

CALCULATING PEAK SNR AND SNR VALUES FOR DIFFERENT FILTERS

```
%Reading original image
noiseless_image = imread('RioDeJaneiro.jpg');
noiseless_image = im2double(noiseless_image);
imshow(noiseless_image);
title('Given noiseless image');
figure;
original_image = imread('RioDeJaneiro_noise.jpg');
noisy_image = im2double(original_image);
imshow(noisy_image);
title('Given noisy image');
%NOISELESS IMAGE SNR
[thisPeaksnr,thisSnr] = psnr(noiseless_image, noiseless_image);
fprintf('\n Original image - Peak-SNR value is : %0.4f', thisPeaksnr);
fprintf('\n Original image - SNR value is : %0.4f\n', thisSnr);
figure;

%METHOD 1: Wiener filter
wiener_filtered = wiener2(noisy_image, [10 10]);
[thisPeaksnr,thisSnr] = psnr(wiener_filtered, noiseless_image);
imshow(wiener_filtered);
title('Wiener filter');
fprintf('\n Wiener filter - Peak-SNR value is : %0.4f', thisPeaksnr);
fprintf('\n Wiener filter - SNR value is : %0.4f', thisSnr);
figure;

%METHOD 2: Average filter
Kaverage_filtered = double(filter2(fspecial('average',3),noisy_image)/255);
[thisPeaksnr,thisSnr] = psnr(Kaverage_filtered, noiseless_image);
imshow(Kaverage_filtered);
title('Average filter');
fprintf('\n Average filter - Peak-SNR value is : %0.4f', thisPeaksnr);
fprintf('\n Average filter - SNR value is : %0.4f', thisSnr);
figure;

%METHOD 3: Maximum filter
maximum_filter = ordfilt2(noisy_image, 9, ones(3,3));
[thisPeaksnr,thisSnr] = psnr(maximum_filter, noiseless_image);
imshow(maximum_filter);
title('Maximum filter');
fprintf('\n Maximum filter - Peak-SNR value is : %0.4f', thisPeaksnr);
fprintf('\n Maximum filter - SNR value is : %0.4f', thisSnr);
figure;

%METHOD 4: Median filter
median_filter = ordfilt2(noisy_image, 5, ones(3,3));
[thisPeaksnr,thisSnr] = psnr(median_filter, noiseless_image);
imshow(median_filter);
title('Median filter');
fprintf('\n Median filter - Peak-SNR value is : %0.4f', thisPeaksnr);
fprintf('\n Median filter - SNR value is : %0.4f', thisSnr);
figure;

%METHOD 5: Minimum filter
minimum_filter = ordfilt2(noisy_image,1,ones(3,3));
[thisPeaksnr,thisSnr] = psnr(minimum_filter, noiseless_image);
imshow(minimum_filter);
title('Minimum filter');
fprintf('\n Minimum filter - Peak-SNR value is : %0.4f', thisPeaksnr);
fprintf('\n Minimum filter - SNR value is : %0.4f', thisSnr);
```

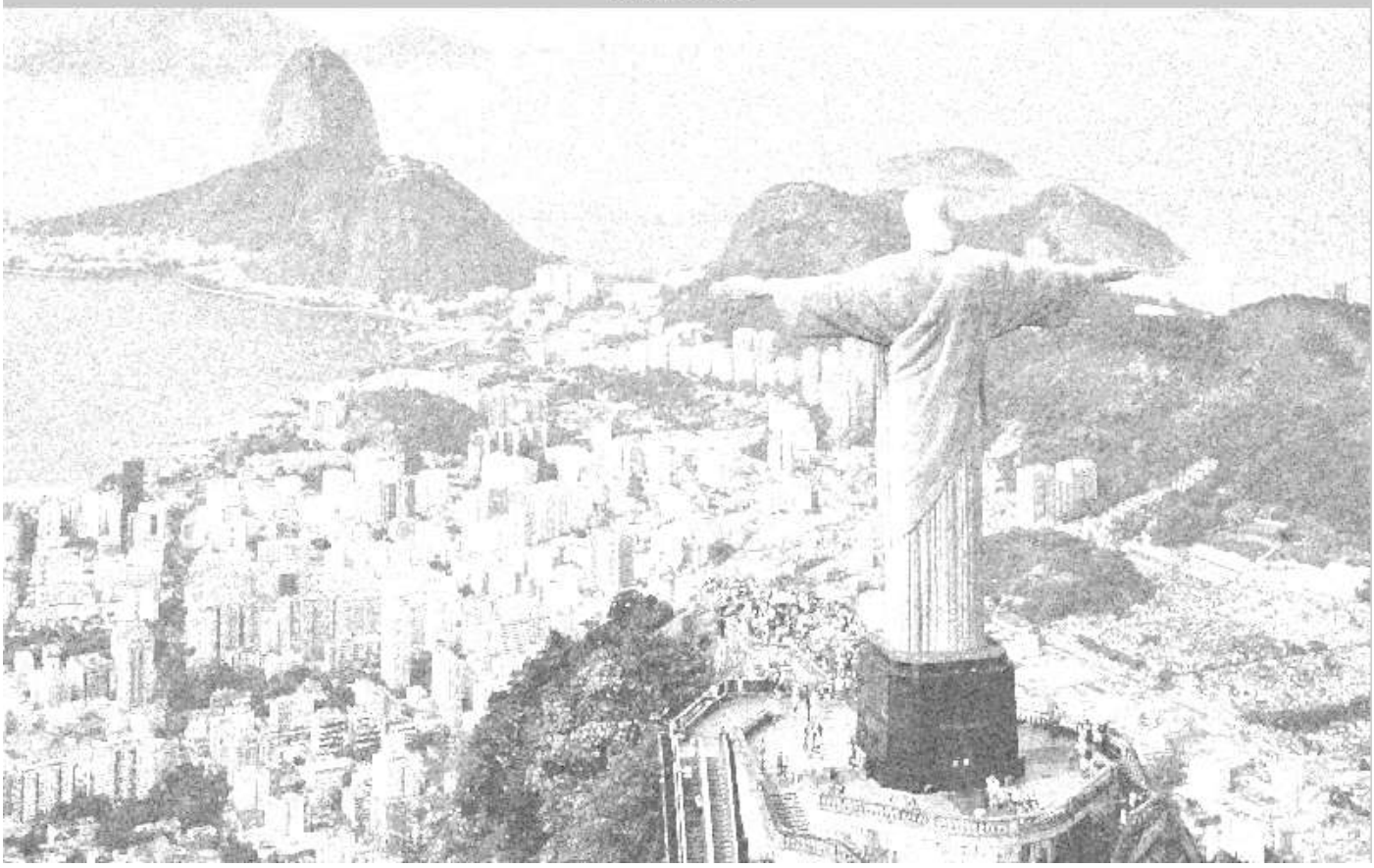
Given noiseless image

Given noisy image

Wiener filter

Average filter

Maximum filter



Median filter


```
Wiener filter - Peak-SNR value is : 11.9548
Wiener filter - SNR value is : 5.7671
```

```
Average filter - Peak-SNR value is : 6.2354
Average filter - SNR value is : 0.0478
```

```
Maximum filter - Peak-SNR value is : 6.8722
Maximum filter - SNR value is : 0.6845
```

```
Original image - Peak-SNR value is : Inf
Original image - SNR value is : Inf
```

```
Median filter - Peak-SNR value is : 11.6533
Median filter - SNR value is : 5.4656
```

```
Minimum filter - Peak-SNR value is : 19.0370
Minimum filter - SNR value is : 12.8494
```

From the definition of SNR (measured in decibels (dB)): an image with low SNR is not as acceptable when compared to an image with high SNR. Hence, we rate the image quality with respect to how high its SNR value is. Original image has SNR value equal to infinity.

It can be seen from the above output peak-snr and snr values that the **quality of filtered images** is as follows:

Minimum filter > (Wiener filter ~ Median filter) > Maximum filter > Average filter

MAX SNR **MIN SNR**