1.5em 0pt

Assignment 3

E9 246: Advance Image Processing
Indian Institute of Science, Bengaluru

Name:	Alokendu Mazumder
SR Number:	20134
Department:	Electrical Engineering
Program:	PhD
Code & Results Link:	**Click Here**

1 MMSE estimate of Laplace source in presence of AWGN

Given a clean image, in presence of gaussian noise, we have to compute the minimum mean squared estimate assuming the source (clean image) is follwing a laplacian distribution. The model defined is as follows:

$$\mathbf{y} = \mathbf{x} + \mathbf{z} \tag{1}$$

Here, $\mathbf{z} \sim \mathcal{N}(0,0.1)$ and $\mathbf{x} \sim L(0,1)$. The MMSE of textitx given y is provided by the mean of the posterior, it can defined as:

$$\hat{x}(y) = \int_{-\infty}^{+\infty} x P_{y|x}(y|x)dx \tag{2}$$

$$= \frac{\int_{-\infty}^{+\infty} x P_{y|x}(y|x) P_x(x)dx}{\int_{-\infty}^{+\infty} P_{y|x}(y|x) P_x(x)dx}$$
$$= \frac{\int_{-\infty}^{+\infty} x P_z(y-x) P_x(x)dx}{\int_{-\infty}^{+\infty} P_z(y-x) P_x(x)dx}$$

Here, P_z & P_x are probability density functions (pdf) of random variables $\mathbf{x}$ & $\mathbf{y}$ respectively. Equation (2) cannot be obtained in closed form with the above distributions being involved, hence we will plot graph for the estimate as a function of y.

1.1 Numerical Results

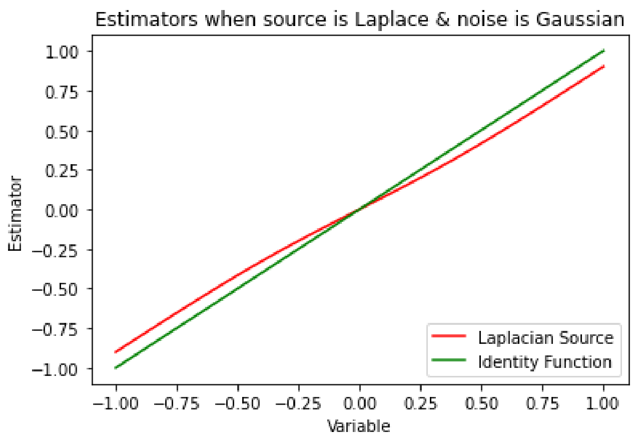


Figure 1: A plot of MMSE when source is Laplacian and Gaussian both separately

The MMSE of laplacian source behaves same way as depicted in the reference paper given with Q1. The slope first increases, then decreases then again increases. To obtain this plot, equation(2) is evaluated numerically using scipy's inbuilt integration function. For each value of y, the integral in equation (2) is computed and plotted.

1.2 Finding MMSE for Laplacian Source

We define the densities as:

$$P_X(x) = \frac{1}{2\sigma_x} e^{\frac{-|x|}{\sigma_x}} \tag{3}$$

$$P_Z(z) = \frac{1}{\sigma_z\sqrt{2\pi}} e^{\frac{-z^2}{2\sigma_z^2}} \tag{4}$$

From (2), we can write the integral as shown: $\hat{x}(y) = \frac{\int_{-\infty}^{+\infty} xP_z(y-x)P_x(x)dx}{\int_{-\infty}^{+\infty} P_z(y-x)P_x(x)dx}$.

We solve the denominator of our estimator first.

$$\int_{-\infty}^{+\infty} \frac{1}{2\sigma_x} e^{\frac{-|x|}{\sigma_x}} \frac{1}{\sigma_z\sqrt{2\pi}} e^{\frac{-(y-x)^2}{2\sigma_z^2}} dx = \frac{1}{2\sigma_x\sigma_z\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{\frac{-|x|}{\sigma_x}} e^{\frac{-(y-x)^2}{2\sigma_z^2}} dx$$

Leaving the constants as of now, we solve the integral:

$$\int_{-\infty}^{+\infty} e^{\frac{-|x|}{\sigma_x}} e^{\frac{-(y-x)^2}{2\sigma_z^2}} dx = \int_{0}^{+\infty} e^{\frac{-(x^2-2xy+y^2)}{2\sigma_z^2} - \frac{x}{\sigma_x}} dx + \int_{-\infty}^{0} e^{\frac{-(x^2+2xy+y^2)}{2\sigma_z^2} + \frac{x}{\sigma_x}} dx$$

Flipping the limits of second integral, we get:

$$= \int_{0}^{+\infty} e^{\frac{-(x^2-2xy+y^2)}{2\sigma_z^2} + \frac{-x}{\sigma_x}} dx + \int_{0}^{+\infty} e^{\frac{-(x^2+2xy+y^2)}{2\sigma_z^2} - \frac{x}{\sigma_x}} dx$$

Putting $x = \sigma_z\sqrt{2}t$, hence $dx = \sigma_z\sqrt{2}dt$, we get:

$$= \int_{0}^{+\infty} e^{-t^2 + \frac{\sqrt{2}y}{\sigma_z}t - \frac{\sigma_z\sqrt{2}}{\sigma_x}} dt + \int_{0}^{+\infty} e^{-t^2 - \frac{\sqrt{2}y}{\sigma_z}t - \frac{\sigma_z\sqrt{2}}{\sigma_x}} dt \tag{5}$$

$$= e^{\frac{-y^2}{\sigma_z^2}} \int_{0}^{+\infty} e^{-t^2 + t(\frac{y\sqrt{2}}{\sigma_z} - \frac{\sigma_z\sqrt{2}}{\sigma_x})} dt + e^{\frac{-y^2}{\sigma_z^2}} \int_{0}^{+\infty} e^{-t^2 - t(\frac{y\sqrt{2}}{\sigma_z} + \frac{\sigma_z\sqrt{2}}{\sigma_x})} dt \tag{6}$$

Before moving forward, we define the complementary error function:

$$erfc(b) = \frac{2}{\sqrt{\pi}} \int_{0}^{+\infty} e^{-x^2-2bx} dx \tag{7}$$

After taking the constants which we left in previous steps, and using (7) to rewrite (6) we get the final form of denominator as:

$$\int_{-\infty}^{+\infty} P_z(y-x)P_x(x)dx = \frac{\sqrt{\pi}}{2} \frac{e^{\frac{-y^2}{2\sigma_z^2}}}{2\sigma_x\sqrt{\pi}} \left\{ erfc\left\{\frac{\sigma_z}{\sqrt{2}\sigma_x} - \frac{y}{\sqrt{2}\sigma_z}\right\} + erfc\left\{\frac{\sigma_z}{\sqrt{2}\sigma_x} + \frac{y}{\sqrt{2}\sigma_z}\right\}\right\} \tag{8}$$

Now, we solve the numerator,

$$\int_{-\infty}^{+\infty} x\frac{1}{2\sigma_x} e^{\frac{-|x|}{\sigma_x}} \frac{1}{\sigma_z\sqrt{2\pi}} e^{\frac{-(y-x)^2}{2\sigma_z^2}} dx = \frac{1}{2\sigma_x\sigma_z\sqrt{2\pi}} \int_{-\infty}^{+\infty} xe^{\frac{-|x|}{\sigma_x}} e^{\frac{-(y-x)^2}{2\sigma_z^2}} dx$$

We perform the similar steps as above and keep the constants outside, we get the integral in this form:

$$= e^{\frac{-y^2}{\sigma_z^2}} \int_{0}^{+\infty} te^{-t^2 + t(\frac{y\sqrt{2}}{\sigma_z} - \frac{\sigma_z\sqrt{2}}{\sigma_x})} dt - e^{\frac{-y^2}{\sigma_z^2}} \int_{0}^{+\infty} te^{-t^2 - t(\frac{y\sqrt{2}}{\sigma_z} + \frac{\sigma_z\sqrt{2}}{\sigma_x})} dt \tag{9}$$

Before solving (9), we define and solve a similar integral as (9), Let:

$$g(b) = \int_{0}^{+\infty} xe^{-x^2-bx} \tag{10}$$

$$= \int_{0}^{+\infty} xe^{-\left(x+\frac{b}{2}\right)^2 + \frac{b^2}{4}} dx \tag{11}$$

Let, $m = x + \frac{b}{2}$, plug it in (10), we get:

$$= \int_{\frac{b}{2}}^{+\infty} \left(m - \frac{b}{2}\right) e^{-m^2+\frac{b^2}{4}} dm \tag{12}$$

$$= -\frac{b}{2} \int_{\frac{b}{2}}^{+\infty} e^{-m^2+\frac{b^2}{4}} dm + \int_{\frac{b}{2}}^{+\infty} m e^{-m^2+\frac{b^2}{4}} dm \tag{13}$$

We can write the second term in addition in (13) as:

$$\int_{\frac{b}{2}}^{+\infty} m e^{-m^2+\frac{b^2}{4}} dm = \int_{0}^{+\infty} e^{-m^2+\frac{b^2}{4}} d\left(\frac{m^2}{2}\right) \tag{14}$$

The final solution of (14) is:

$$\int_{0}^{+\infty} e^{-m^2+\frac{b^2}{4}} d\left(\frac{m^2}{2}\right) = \left[e^{-m^2+\frac{b^2}{4}}\right]_{\frac{b}{2}}^{+\infty} = 1 \tag{15}$$

Now, we solve for the first term in (13) by putting a dummy variable $m = x + \frac{b}{2}$:

$$-\frac{b}{2} \int_{\frac{b}{2}}^{+\infty} e^{-m^2+\frac{b^2}{4}} dm = -\frac{b}{2} \int_{0}^{+\infty} e^{-x^2-bx} dx \tag{16}$$

Now, we replace the variable b in (10) with a dummy variable y and rewrite (10) completely using (16) as:

$$g(y) = \int_{0}^{+\infty} x e^{-x^2-yx} = 1 - \frac{y}{2} \int_{0}^{+\infty} e^{-x^2-yx} dx = 1 - \frac{y}{2}\frac{\sqrt{\pi}}{2} erfc\left(\frac{y}{2}\right) \tag{17}$$

Using (16) we can rewrite (9) as:

$$= e^{\frac{-y^2}{\sigma_z^2}} \left\{1 - \frac{y}{2}\frac{\sqrt{\pi}}{2} erfc\left(\frac{\sigma_z}{\sqrt{2}\sigma_x} - \frac{y}{\sqrt{2}\sigma_z}\right)\right\} - e^{\frac{-y^2}{\sigma_z^2}} \left\{1 - \frac{y}{2}\frac{\sqrt{\pi}}{2} erfc\left(\frac{\sigma_z}{\sqrt{2}\sigma_x} + \frac{y}{\sqrt{2}\sigma_z}\right)\right\} \tag{18}$$

Now, rearranging (18) and taking all the constants that we neglected earlier during derivation, we finally write the numerator as:

$$\int_{-\infty}^{+\infty} x P_z(y-x) P_x(x) dx = \frac{\sqrt{\pi}}{2} \frac{e^{\frac{-y^2}{2\sigma_z^2}}}{2\sigma_x\sigma_z\sqrt{2\pi}} \left\{\frac{y}{2} erfc\left(\frac{\sigma_z}{\sqrt{2}\sigma_x} + \frac{y}{\sqrt{2}\sigma_z}\right) - \frac{y}{2} erfc\left(\frac{\sigma_z}{\sqrt{2}\sigma_x} - \frac{y}{\sqrt{2}\sigma_z}\right)\right\} \tag{19}$$

By dividing (19) with (8) and cancelling all the common terms in numerator & denominator, we get our final estimator equation as:

$$\hat{x}(y) = \frac{y}{2\sigma_z\sqrt{\pi}} \frac{erfc\left(\frac{\sigma_z}{\sqrt{2}\sigma_x} + \frac{y}{\sqrt{2}\sigma_z}\right) - erfc\left(\frac{\sigma_z}{\sqrt{2}\sigma_x} - \frac{y}{\sqrt{2}\sigma_z}\right)}{erfc\left(\frac{\sigma_z}{\sqrt{2}\sigma_x} + \frac{y}{\sqrt{2}\sigma_z}\right) + erfc\left(\frac{\sigma_z}{\sqrt{2}\sigma_x} - \frac{y}{\sqrt{2}\sigma_z}\right)} \tag{20}$$

2 Image Denoising

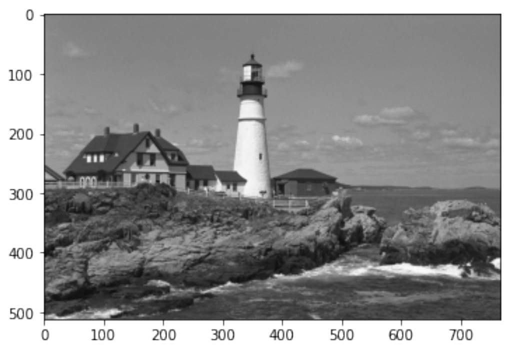
We use the same symbols and notations from above. We have a clean image which is being corrupted by Gaussian noise. it can be modelled as:

$$\mathbf{y} = \mathbf{x} + \mathbf{z} \tag{21}$$

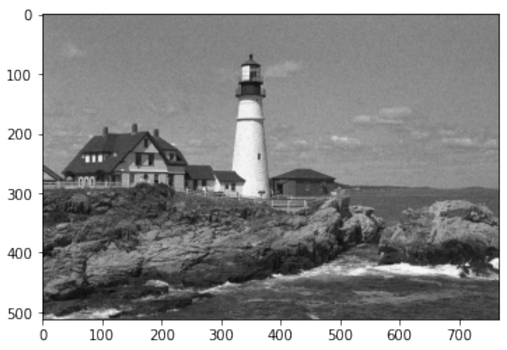
here, $\mathbf{z} \sim \mathcal{N}(0{,}100)$. Cv2's *GaussinBlur*() is used to get low pass information, *mean_squared_error*() from sklearn's metric is used to get mean squared error. Noise matrix is generated using numpy's *random.normal*() with specifications given in the question. For convolution, scipy's *convolve2d*() is used.

2.1 Low pass Gaussian filtering

Given a set of Gaussian kernel sizes and it's standard deviations, corrupted image is being blurred with various kernel & standard deviation combination, and mean squared error is computed w.r.t original clean image. A function $minmse_fit()$ is developed which will spit out the kernel size and standard deviation for which mean squared error is minimum.



(a) Clean image



(b) Corrupted image with gaussian noise

Figure 2: Input images for denoising

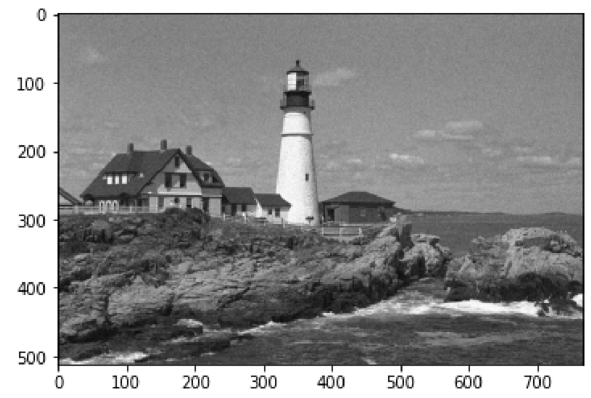
2.1.1 Observation

- The best filter configuration which gives minimum MSE is achieved at $kernel = (3,3)$ & $\sigma_{LPF} = 1$.
- If variance is increased, the image will get too blurred, hence it will increase the MSE.
- Also, with a little variance of LPF say 0.1, denoising won't take place effectively as the gaussian filter won't be able to perform effective smoothing, this will also result in increase of MSE.
- It can be observed that, as kernel size increases, the MSE also increases for high variance values. For low variance, MSE is not very bad.

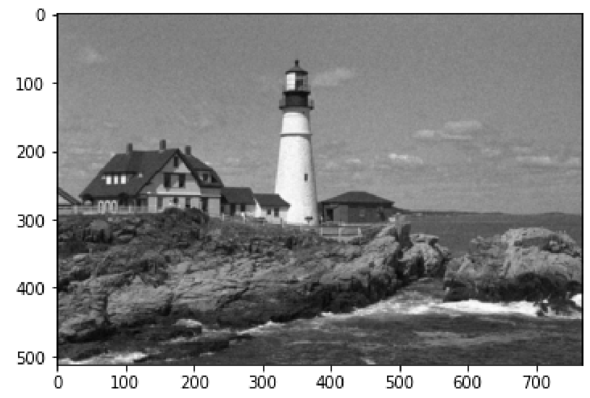
Table 1: MSE table for Gaussian filter based denoising

MSE (Kernel Size with σ^2_{LPF})	**Var=0.01**	**Var = 1**	**Var=4**	**Var=16**	**Var=64**
Kernel $(3,3)$	99.63	89.47	110.11	115.47	116.81
Kernel $(7,7)$	99.63	114.24	214.82	261.39	274.41
Kernel $(11,11)$	99.63	114.28	229.75	318.38	350.56

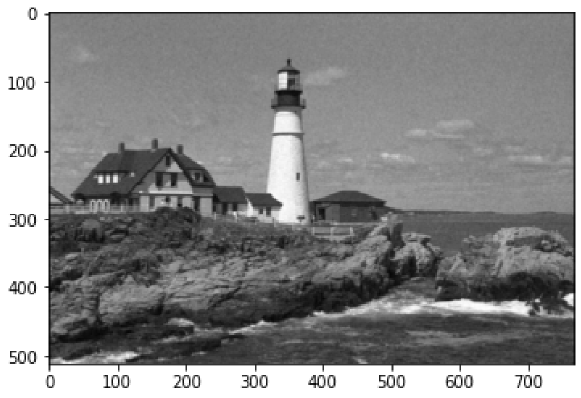
2.1.2 Results



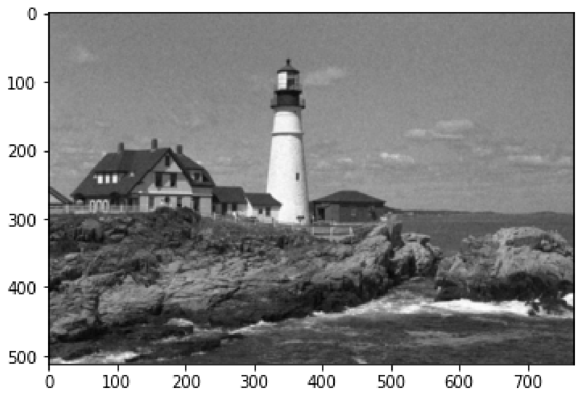
(a) Output with $\sigma_{LPF} = 0.1$



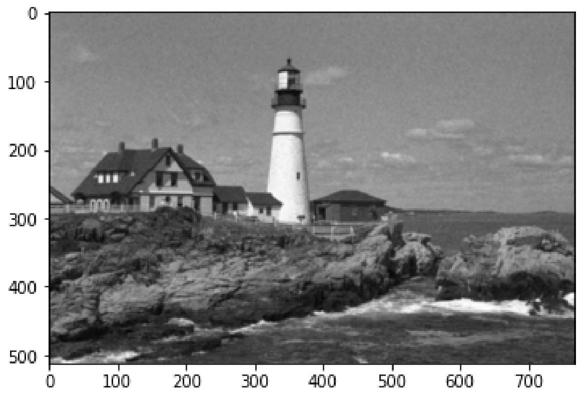
(b) Output with $\sigma_{LPF} = 1$



(c) Output with $\sigma_{LPF} = 2$

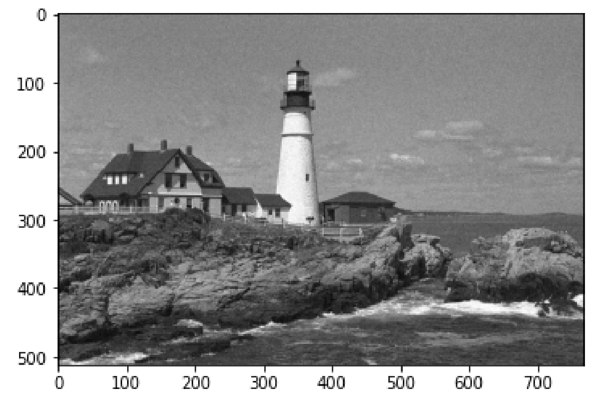


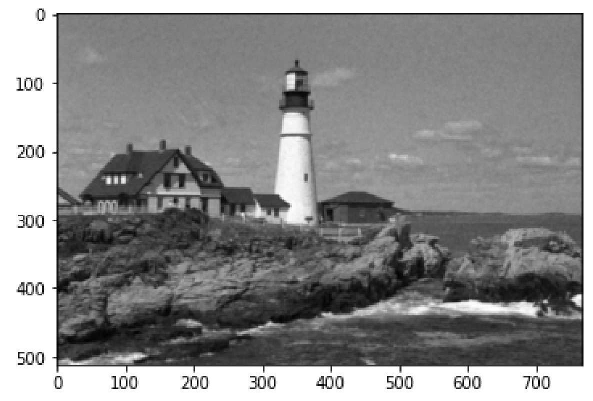
(d) Output with $\sigma_{LPF} = 4$

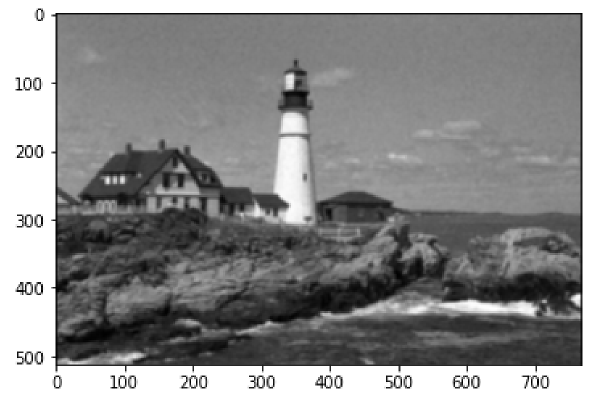


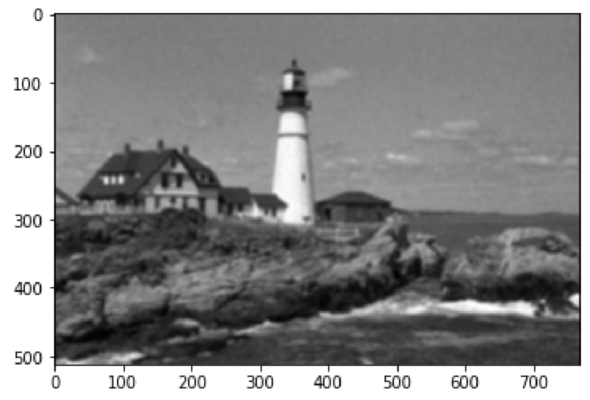
(e) Output with $\sigma_{LPF} = 8$

Figure 3: Results for kernel size $(3, 3)$


(a) Output with $\sigma_{LPF} = 0.1$


(b) Output with $\sigma_{LPF} = 1$


(c) Output with $\sigma_{LPF} = 2$


(d) Output with $\sigma_{LPF} = 4$

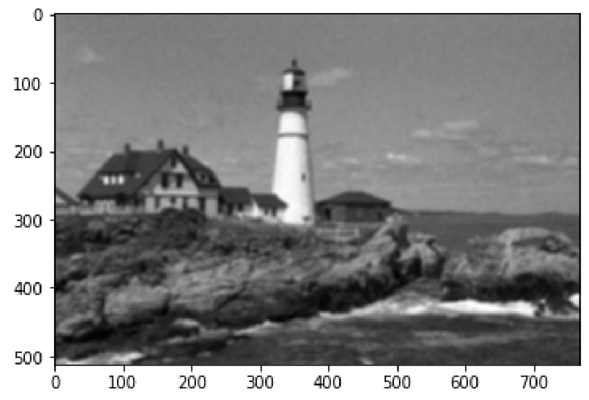
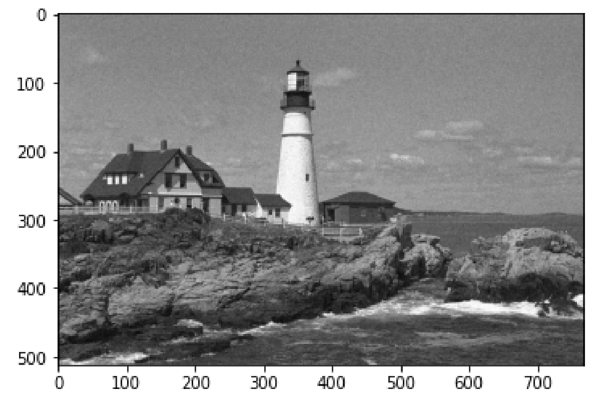
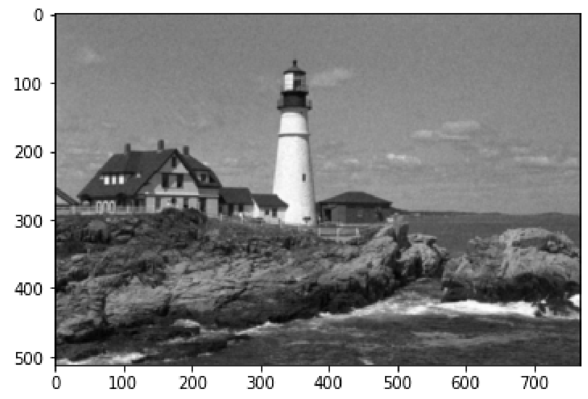

(e) Output with $\sigma_{LPF} = 8$

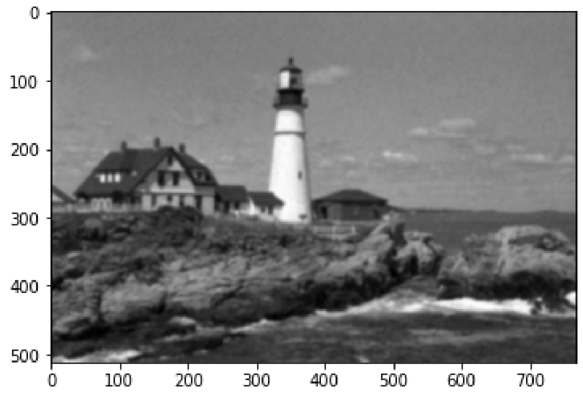
Figure 4: Results for kernel size $(7, 7)$



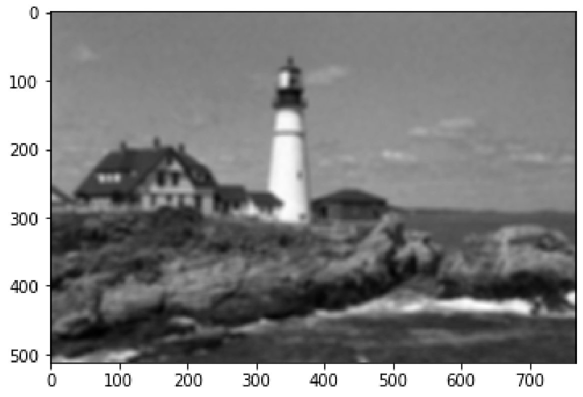
(a) Output with $\sigma_{LPF} = 0.1$



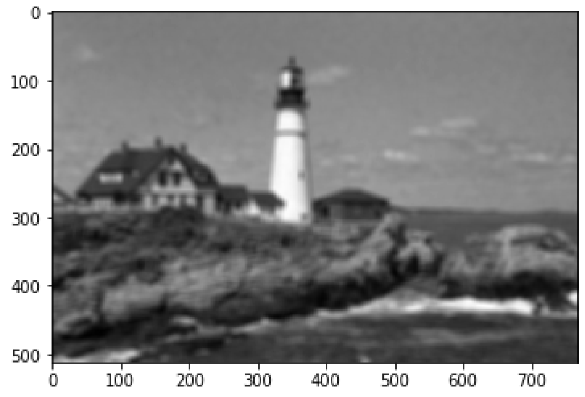
(b) Output with $\sigma_{LPF} = 1$



(c) Output with $\sigma_{LPF} = 2$



(d) Output with $\sigma_{LPF} = 4$



(e) Output with $\sigma_{LPF} = 8$

Figure 5: Results for kernel size $(11, 11)$

2.2 MMSE Filtering

We use the same symbols and notations from abovee. From equation (3) of our model, assuming the source/clean image to follow gaussian distribution with gaussian noise, the MMSE estimate is defined as:

$$\hat{x} = \mu_y + \frac{\sigma_{x_1}^2}{\sigma_{x_1}^2 + \sigma_{z_1}^2} * y_1 \tag{22}$$

Here, y_1 denotes high pass version of corrupted image,where $\mathbf{y}$ is the corrupted image with gaussian noise with zero mean and 100 variance, and μ_y is low pass filter output of $\mathbf{y}$. The reconstructed image using MMSE is denoted by $\hat{x}$ & $\sigma_{x_1}^2$ & $\sigma_{z_1}^2$ are variances of high pass versions of clean image and noise respectively. The best filter from previous part is used to get the low pass version of clean image, corrupted image and noise.

2.2.1 Variance Computation

For, clean image $\mathbf{x}$, corrupted image $\mathbf{y}$ and noise $\mathbf{z}$, we first get their low pass versions and then substract it with original counterpart to get high pass versions.

$$\begin{aligned} x_1 &= \mathbf{x} - \mu_x \\ y_1 &= \mathbf{y} - \mu_y \\ z_1 &= \mathbf{z} - \mu_z \end{aligned}$$

Here, μ_x, μ_y, μ_z are low pass versions of $\mathbf{x}$, $\mathbf{y}$ & $\mathbf{z}$ respectively and x_1, y_1 & z_1 is their high pass versions respectively. The variance of y_1 can be computed straight forward from the corrupted image itself, like this:

$$\sigma_{y_1}^2 = \frac{1}{MN} \sum_{\forall k} \sum_{\forall l} [y(k,l) - \mu_y]^2 \tag{23}$$

Now, if $\mathbf{z}$ has zero mean then z_1 will also have zero mean, but the variance of both may not be same. The z_1 can be written as:

$$z_1 = z(i,j)(1 - h(0,0)) + \sum_{k \neq 0} \sum_{l \neq 0} h(k,l) z(i+k, j+l) \tag{24}$$

The LPF kernel here is denoted by $h(,)$. From the above equation, the variance of high pass version of the noise can be computed as:

$$\sigma_{z_1}^2 = [(1 - h(0,0))^2 + \sum_{k \neq 0} \sum_{l \neq 0} h(k,l)^2] * \sigma_z^2 \tag{25}$$

The same formulae above is used in code also. Now from (5) & (7), we can find the variance of high pass version of clean image as:

$$\sigma_{x1}^2 = \sigma_{y1}^2 - \sigma_{z1}^2 \tag{26}$$

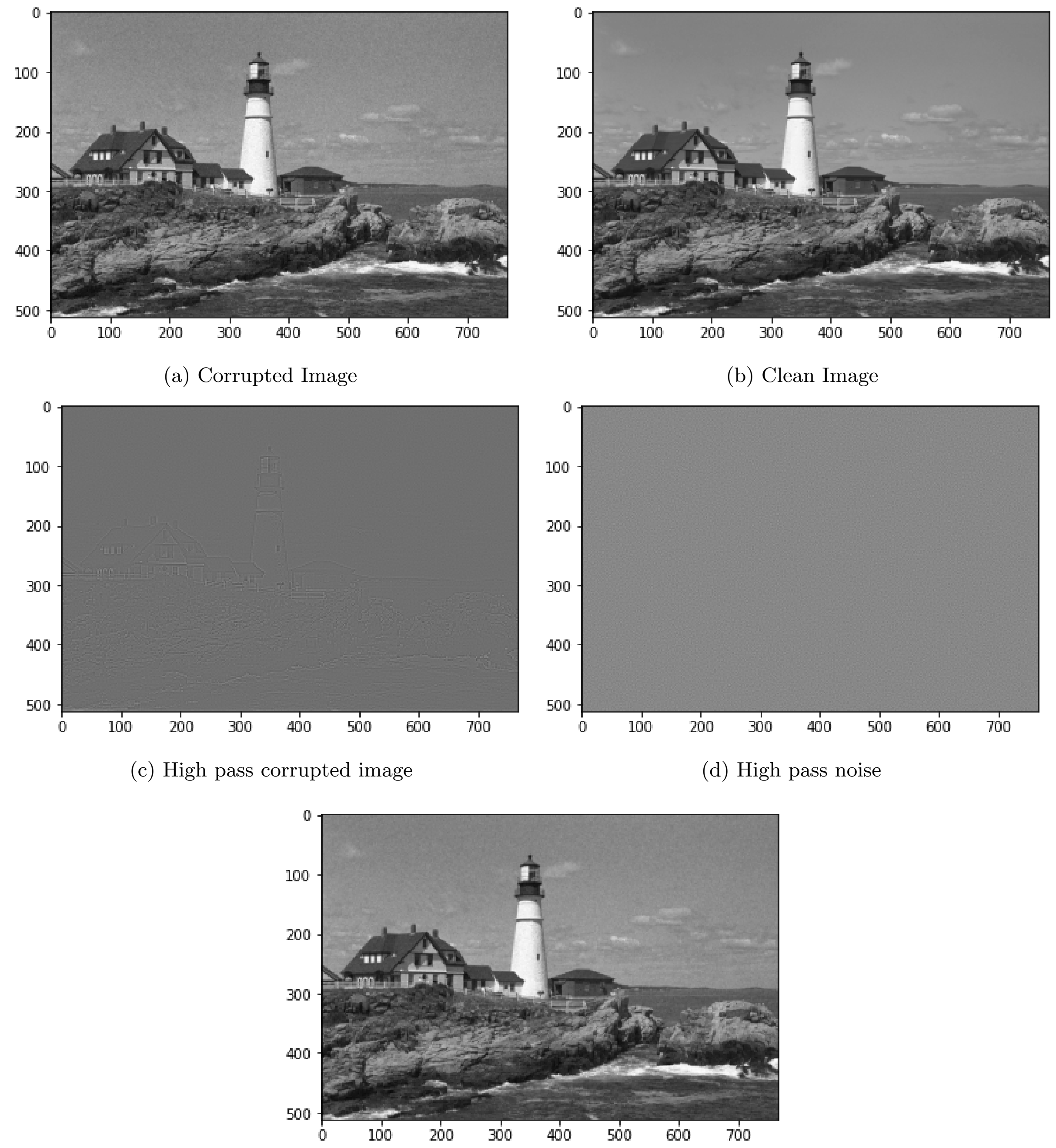
2.2.2 Observations

The mean squared error between clean image and MMSE image came out to be **60.19**. The model of input being Gaussian distributed might result in inaccurate estimate of corrupted image variance i.e $\sigma_{y_1}^2$. This might provide poor results when the corrupting noise has too high variance.

- Variance of high pass version of corrupted image ($\sigma_{y_1}^2$) = **154.79**
- Variance of high pass version of noise ($\sigma_{z_1}^2$) = **65.58**
- Variance of high pass version of clean image ($\sigma_{x_1}^2$) = **89.20**

2.2.3 Results

:



(a) Corrupted Image

(b) Clean Image

(c) High pass corrupted image

(d) High pass noise

(e) MMSE restored image

Figure 6: Images generated in various parts of MMSE process

2.3 Adaptive MMSE

We use the same symbols and notations from above. In Adaptive MMSE, rather than performing estimation over the whole image, we take small patches and perform MMSE over them. It depends on the local properties of the corrupted image. Along with the patch size we choose a stride, i.e by how much one needs to skip while moving side by side or up and down. The pixels belonging to overlapping patches are averaged out later. The denoising model will share same structure as (22), but the modified equation for adaptive MMSE is:

$$\hat{x}_{patch} = \mu_y + AdapMMSE(y_1, z) \tag{27}$$

2.3.1 Algorithm of Adaptive MMSE

We define the algorithm to find $AdapMMSE()$ as:

Algorithm 1 Algorithm to estimate the high pass version of the patches

```
1: procedure ADAPMMSE($y_1, z$)                                        ▷ This is a test
2:     $\hat{x} \leftarrow [0]$                          ▷ Declare an array full of zeros with same size as $y_1$
3:     overlap $\leftarrow \hat{x}$
4:     stride $\leftarrow 6$
5:     Patch_Size $\leftarrow 11$
6:     $\sigma_2^2 \leftarrow$ findvariance($z$)                                 ▷ Variance of the noise
7:     for $i \leftarrow 0$ to $(y_1.shape[0] - 3)$ do
8:         for $j \leftarrow 0$ to $(y_1.shape[1] - 7)$ do
9:             dummy $\leftarrow y_1[i : i +$ Patch_Size$, j : j +$ Patch_Size$]$          ▷ Choosing patches
10:            $\sigma_1^2 \leftarrow$ findvariance(dummy)          ▷ Computing variance of the chosen patch
11:            $\hat{x}[i : i +$ Patch_Size$, j : j +$ Patch_Size$] \leftarrow \left(1 - \frac{\sigma_2^2}{\sigma_1^2}\right) *$ dummy     ▷ Estimating the high pass version of the patch
12:            overlap$[i : i +$ Patch_Size$, j : j +$ Patch_Size$] + \leftarrow 1$ ▷ Updating the pixel locations of overlap
13:        end for
14:    end for
15:    return $\hat{x}$/overlap     ▷ Returning the estimate with average being performed for overlapping regions
16: end procedure
```

Here, $AdapMMSE()$ computes the high pass estimate for every patch of high pass version of the noisy image, then it is added with the low pass version of the noisy image to get the final output.

2.3.2 Findings & Results

The mean squared error of the estimated image with original image came out to be **37.55**, which is very low as compared to the MSE obtained in **Q(2)(b)**, which is **60.19**. Low mean squared error was expected as we

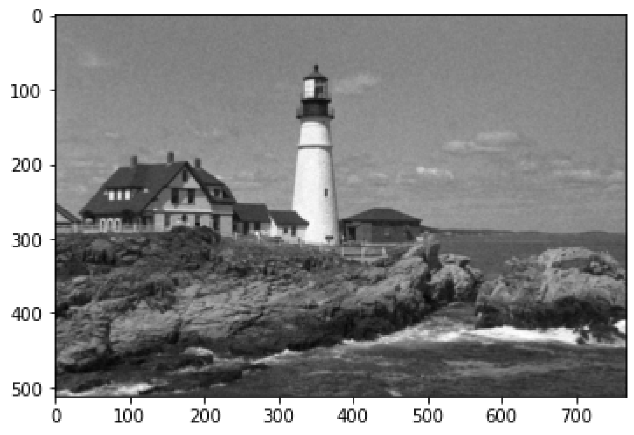


Figure 7: Adaptive MMSE result

are denoising the image locally unlike in **Q(2)(b)**. As we are computing the variance of y_1 for each patch, we are denoising based on the *favour* of each patch separately, hence getting a better result than global MMSE.

2.4 Comparison of MSE score in Q2 (a), (b), (c)

Table 2: MSE Table for various methods used in Q(2)

MSE v/s Methods	**Gaussian Blur**	**MMSE**	**Adaptive MMSE**
MSE	89.47	60.19	37.55

Comments: We saw that adaptive MMSE performs best among all methods, reason being it reconstructs images using a local patch, so it's estimate is locally favourable.

3 Image Sharpening

We use the same symbols and notations from above. In image sharpening, we split the image into high pass and low pass versions, multiply the high pass version with a constant gain parameter and club the low pass and enhanced high pass version. This technique usually increases the edge strength/high pass strength. Using the notations used earlier, we can write the process as:

$$y_{enhanced} = \mu_y + \lambda y_1 \tag{28}$$

3.1 Image sharpening with constant gain

I have used the following pipeline, given y_1, First it is convolved with the best LPF from Q(2)(a). This output is then fed to a high pass filter with kernel given. The HPF output is multiplied by a constant gain factor λ. The final HPF out is then added with the LPF one to get the sharpened image. The block diagram of my pipeline is:

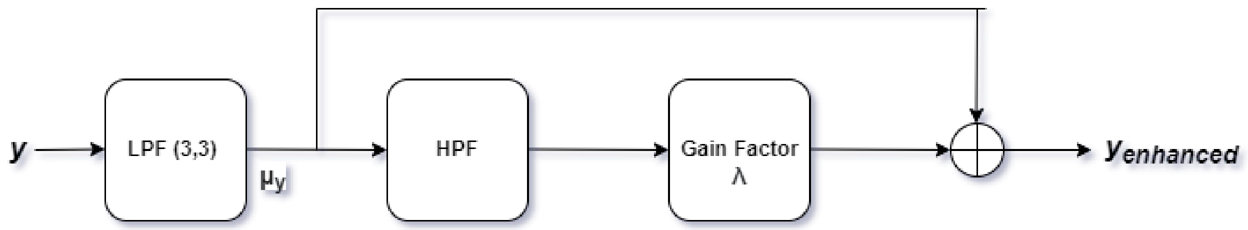


Figure 8: Image sharpening pipeline

3.1.1 Image sharpening with given kernel

The high pass kernel is given by:

$$\mathbf{H}_{hpf} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{29}$$

For the above setup, our task is to find the optimal λ for which MSE is minimized. So, to find the optimal λ I have solved the following optimization problem:

$$\lambda^* = \arg\min_{\lambda} \sum_i \sum_j [\mathbf{x}(i,j) - (\mu_y(i,j) + \lambda y_H(i,j))]^2 \tag{30}$$

Here, y_H is the is the high pass version of μ_y, i.e output of the second block of pipeline.

$$\frac{d}{d\lambda} \sum_i \sum_j [\mathbf{x}(i,j) - (\mu_y(i,j) + \lambda y_1(i,j))]^2 = 0$$

$$\sum_i \sum_j x(i,j) * y_H(i,j) - \sum_i \sum_j (\mu_y(i,j) + \lambda y_H(i,j)) * y_H(i,j) = 0$$

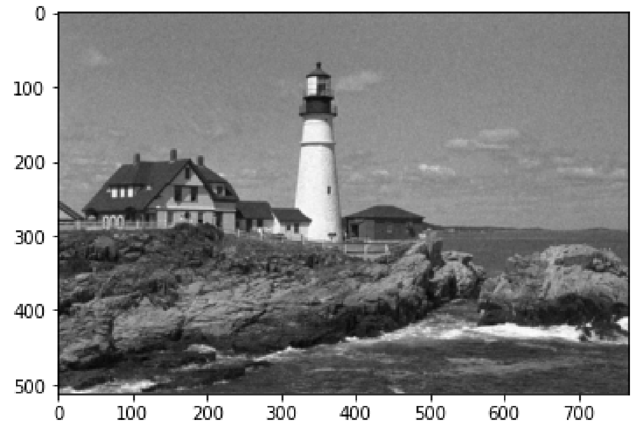
$$\sum_i \sum_j \mu_y(i,j) * y_H(i,j) + \lambda \sum_i \sum_j y_H^2(i,j) = \sum_i \sum_j x(i,j) * y_H(i,j)$$

Solving for λ in above steps, we get:

$$\lambda^* = \frac{\sum_i \sum_j (\mu_y(i,j) - x(i,j)) * y_H(i,j)}{\sum_i \sum_j y_H^2(i,j)} \tag{31}$$

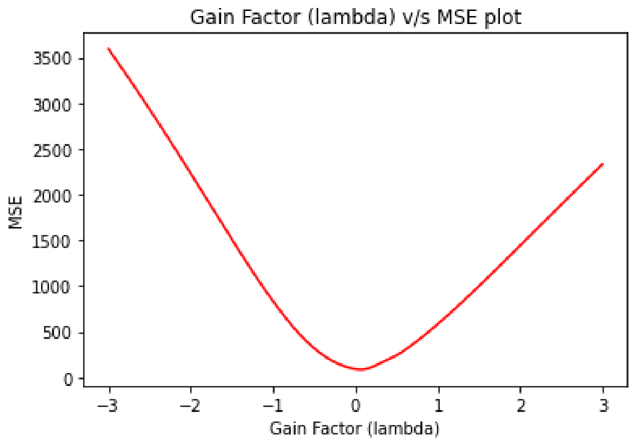
I have implemented this question in two parts, one where I have used to optimal λ from (12) and computed the MSE. In the other one, I have declared a line of λ in the interval $(-3, 3)$ with 1000 samples and for each λ I have generated a sharpened image and computed it's MSE w.r.t original clean image. For the given set of images, optimal λ using (12) came out to be **0.077** (In graph also it's somewhere around this value) and MSE at optimal λ is **83.86**.

3.1.2 Results



(a) Image with optimal λ

(b) Clean Image



(c) Plot of λ v/s MSE

Figure 9: Images generated in various parts Q(3)(a)

3.1.3 Observation

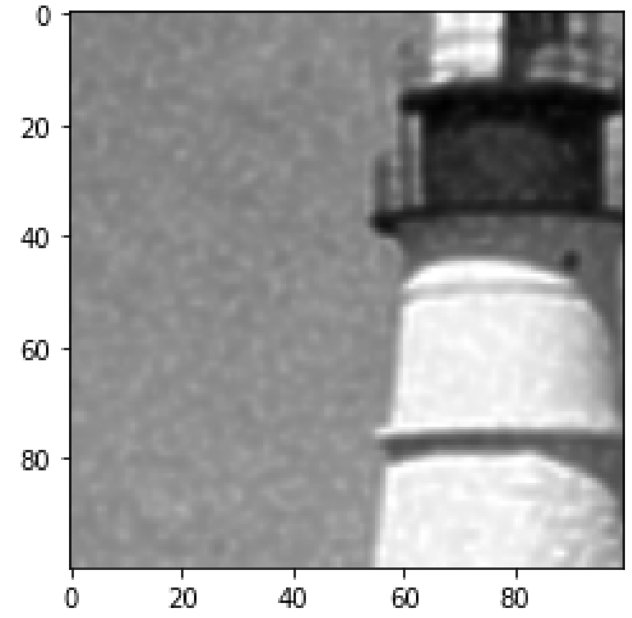
- We see that MSE is generally poor for largely negative or largely positive values of λ.Performance is best for values in [0,1].
- As $|\lambda|$ increases above 0.05, we see that, since the HPF image is almost zero mean and consisting of both positive and negative values, it causes the output to cross intensity limits of 0 and 255. As a result, when pixel intensities are saturated to these values, the MSE worsens severely.

In generating sharpened images using λ that belongs to $(-3, 3)$ we found that the optimal λ is around **0.063** (In graph also it's somewhere around this value). It is in-fact very close to the optimal value, the reason it's not the exact is because, here, λ is generated using $numpy's\ linspace()$ method. So it is possible that here λ is unable to take the real optimal value i.e **0.077** and hence we get a value which is close to it and generated by $linspace()$.

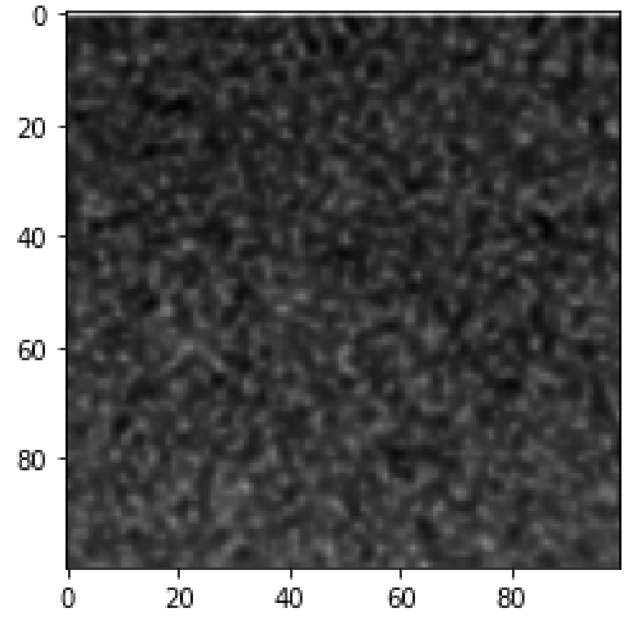
3.2 Perceptual quality of sharpened image

In this section we will analyse the quality of sharpened image w.r.t different gain parameter, and comment on weather a smaller gain, larger gain or the same gain would have been the best for the perceived quality for each region.

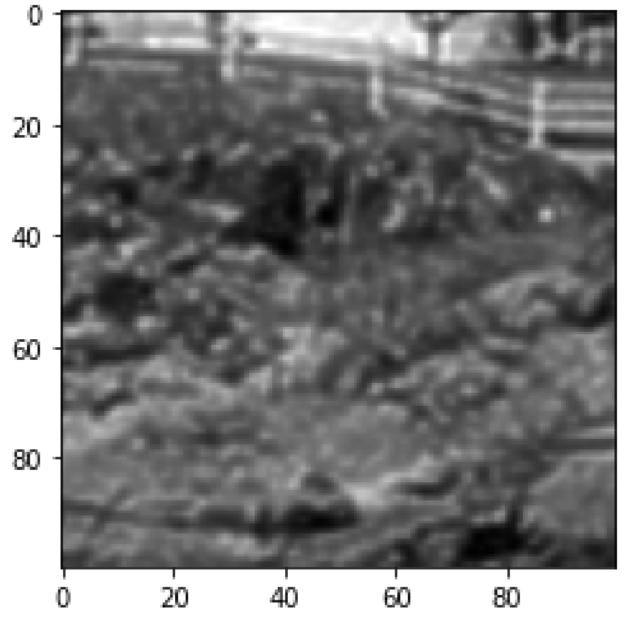
3.2.1 Results



(a) Edge region

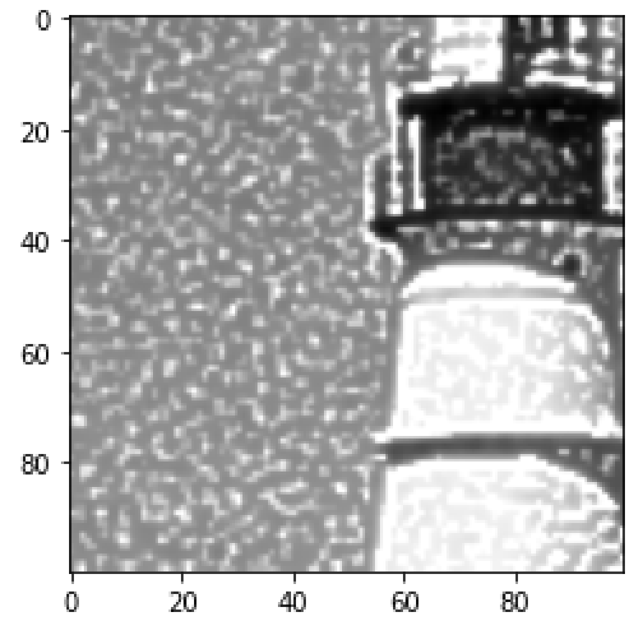


(b) Smooth region

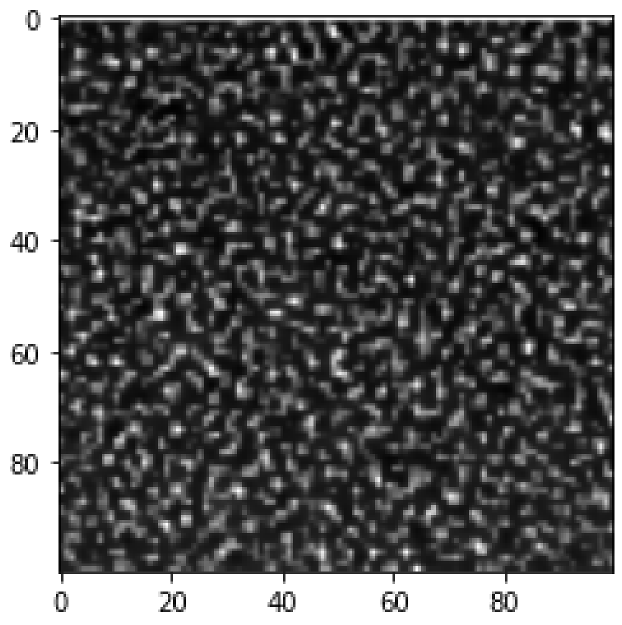


(c) Texture region

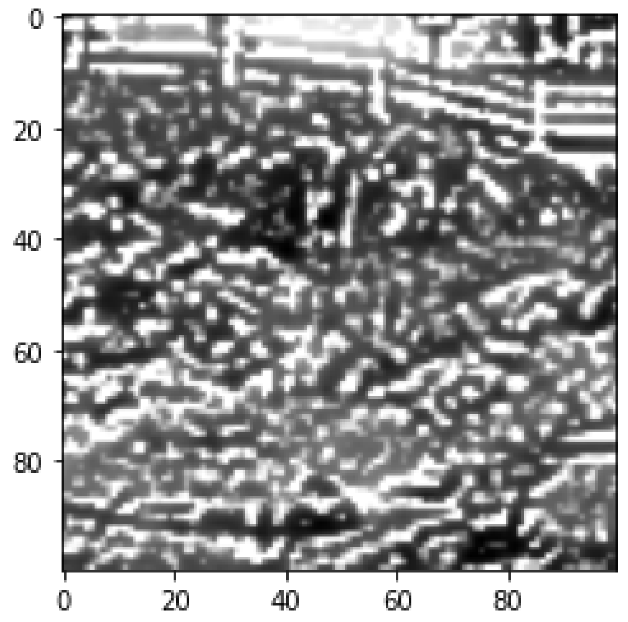
Figure 10: These are images for $\lambda = 0.1$



(a) Edge region

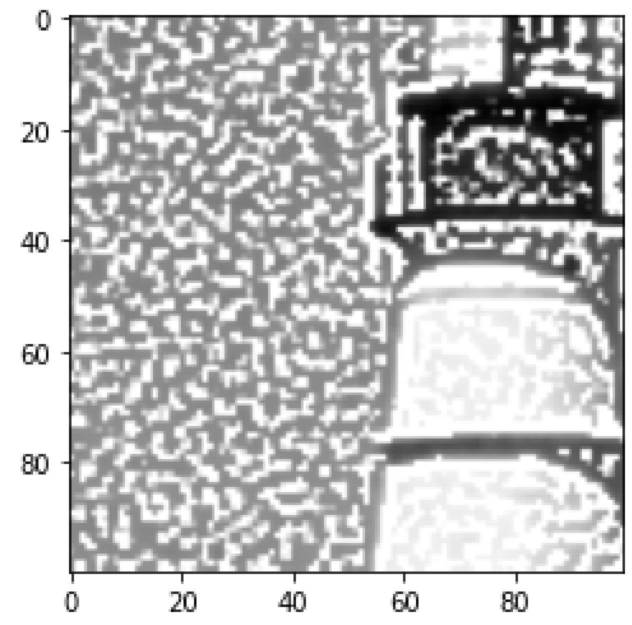


(b) Smooth region

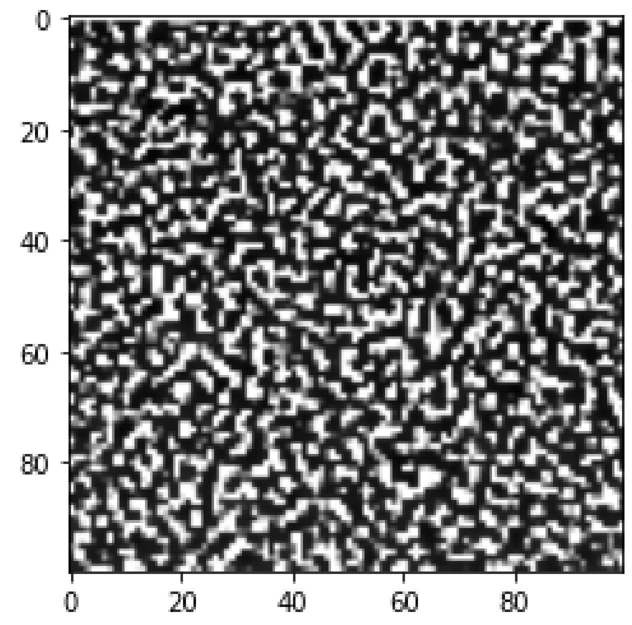


(c) Texture region

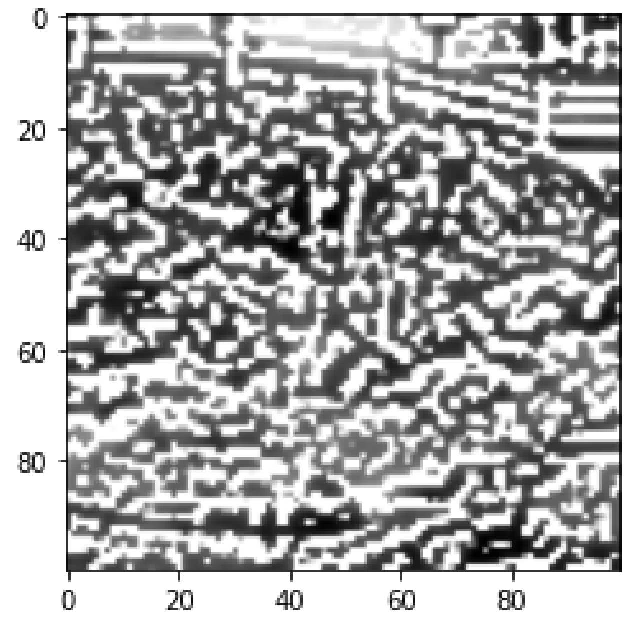
Figure 11: These are images for $\lambda = 2.5$



(a) Edge region



(b) Smooth region



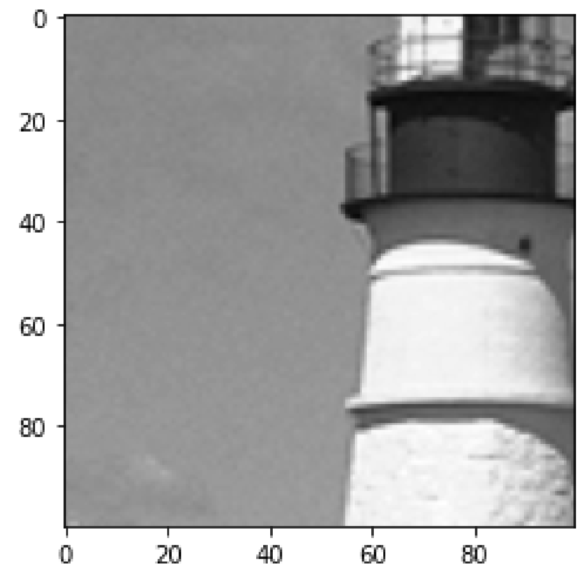
(c) Texture region

Figure 12: These are images for $\lambda = 9$

3.2.2 Observations

- From the images, we can clearly say that the smooth regions looks good for lower values of λ, as gain increases we can see presence of white spots due to noise on smooth regions.
- The edges looks good for moderate/same values of λ, for low value of gain, the output is bit blurred and hence edges are not prominent, for high gain we can see some distortions within the edges.
- And the texture can be *identified* nicely for high values of λ,for low gain, images is blurred, hence textures are not prominent, for moderate values of gain, they started to appear, and they are more prominent at high gain values.

Comment: The sharp regions do look good visually as we can spot the presence of white regions around edges.


(a) Edge in clean image

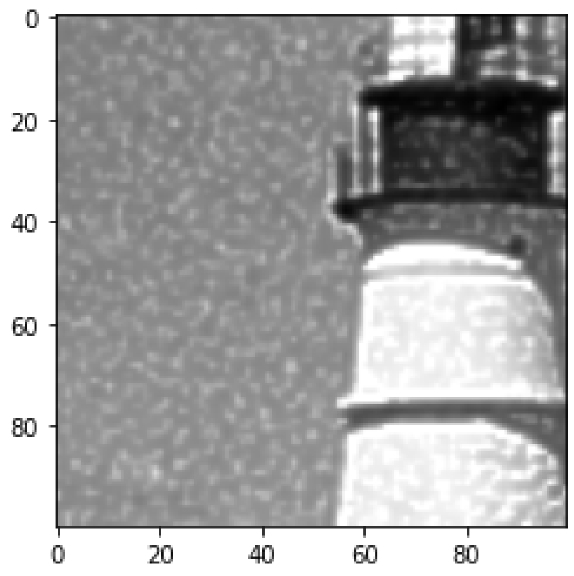

(b) Clean Image

Figure 13: Edge comparison