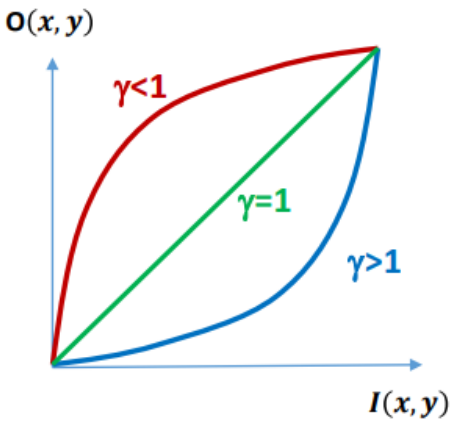
Report

1. **Low-Luminosity Enhancement:**

Method1: Power-Law Transformation

$$O(x,y) = I_{max}(\frac{I(x,y)}{I_{max}})^{\gamma}$$

I choose $\gamma < 1$ to meet the enhancement effect. From the formula, I_{max} is 255 for 8 bits per-channel representation. If gamma's value less than one, the low light part will be mapping to a wide range of values. Hence, the low light part details on the original image will be shown up. However, the bright part will be mapping to a narrow range of values. Hence, the bright part details on the original image will be hidden.

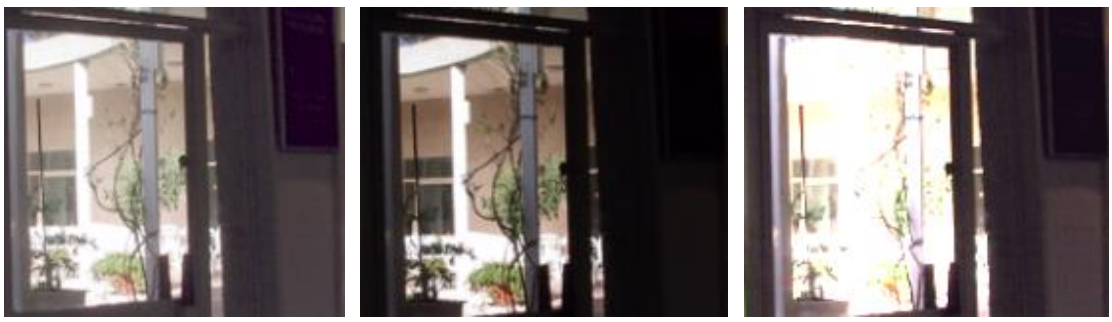


Method2: Bright and Contrast Adjustment

$$g(i,j) = \alpha \cdot f(i,j) + \beta$$

From the formula, the f(i, j) is the term of the original image at the i-th row and j-th column, the α and β are so called gain and bias. This operator is a point operator, where beta is a constant value adds up to every pixel, and those values outside of the [0, 255] range will be clamped to 0 or 255. It can be seen that this method will improve the brightness, but the bright part will become brighter and even saturated.

Comparison (Original image)	Method1 (Output image)	Method2 (Output image)
Bright part	Less colored	Brighter
Low light part	Brighter	Brighter



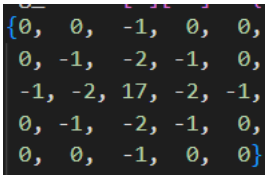
Method1 Input Method2

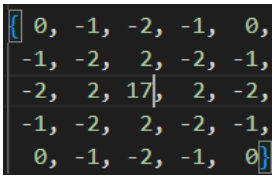
2. **Sharpness Enhancement:**

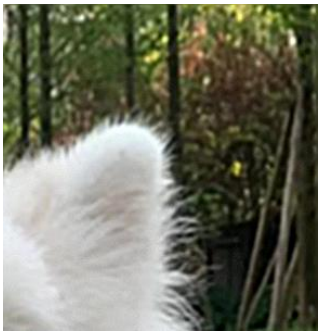
In this section, I use Laplacian Filter to detect the edges of the image. The Laplacian filter enhances edges by boosting the contrast between adjacent pixels, more clearly, it enhances the area of rapidly intensity change, therefore, making edges (high frequency detail) more prominent.

Laplacian Gaussian Filtering Convolution

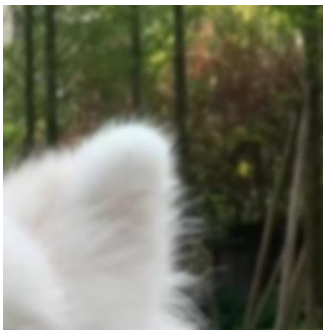
```
    Input: Original image of shape [H, W, C]
    Output: Sharpen image with respect to the original image
1   Initialize: assign zero to y and x
2   N: Kernel size of the filter
3   while (y < H) do
4       while (x < W) do // every (x, y) with C size of data
5           If x or y at edge do
6               assign original image data to the output
7           else do
8               Convolution loop with Laplacian gaussian kernel
9               Clamp the values inside [0, 255] and assign to output
10          Increase x by 1
11      end
12      Increase y by 1
13  end
```







Kernel1



Input

Kernel2

From the figures, we can see that the kernel one shows the effect between input and kernel two. The edges like the fur of the dog become sharper and the background becomes clearly.

3. **Image Denoise:**

Method1: Gaussian Smoothing

From the formula, the σ_X and σ_Y can be the same. The Gaussian kernel have some properties:

$$h(x,y) = \frac{1}{\sqrt{2\pi\sigma_x^2}\sqrt{2\pi\sigma_Y^2}} e^{-(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2})}$$

1. The center of the kernel has the higher weight, whereas the values far away from the center have the lower weights.
2. If the sigma becomes larger, the values of the kernel beside the center will get closer to the center value. On the other hand, the sigma becomes smaller, the values beside the center will get far away from the center.

From the second property, if the we set the sigma value higher, the smoothing effect will be more obvious. However, the image will be more blurred because the kernel is closed to a moving average filter (simply a low-pass filter).

$\sigma = 3.0, \text{kernel} = 5$

$\sigma = 3.0, \text{kernel} = 11$

$\sigma = 10.0, \text{kernel} = 5$

$\sigma = 10.0, \text{kernel} = 11$

We can see that the larger the sigma value or kernel size, the effect of denoising becomes more obvious, however, the image becomes more blurred.

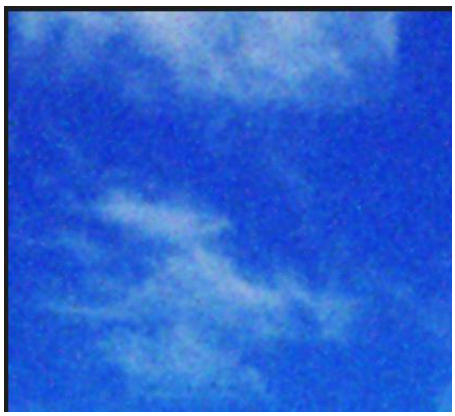
Method2: Bilateral Filtering

$$k(\mathbf{x}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} c(\xi,\mathbf{x})s\big(\mathbf{f}(\xi),\mathbf{f}(\mathbf{x})\big)d\xi$$

$$c(\xi,\mathbf{x}) = \exp\{-\frac{1}{2}(\frac{d(\xi,\mathbf{x})}{\sigma_d})^2\} \quad \text{where} \quad d(\xi,\mathbf{x}) = \|\xi-\mathbf{x}\|$$

$$s(\xi,\mathbf{x}) = \exp\{-\frac{1}{2}(\frac{\delta(f(\xi),f(\mathbf{x}))}{\sigma_r})^2\} \quad \text{where} \quad \delta(\phi,\mathbf{f}) = \|\phi-\mathbf{f}\|$$

From the formula above, $\mathrm{k(x)}$ stands for the Bilateral kernel; $\mathrm{c(\xi,x)}$ is the spatial Gaussian kernel, which depends on the spatial distance of the ξ and x; $\mathrm{s\big(f(\xi),f(x)\big)}$ is called the range Gaussian kernel, which depends on the absolute difference in intensity values between $\mathrm{f(\xi)}\ and\ f(x)$. The x in the formula is the current computing point at the kernel center. The Bilateral filter considers both the spatial proximity and intensity similarity of pixels when determine the influence on the current computing point. Pixels that are closed to the target pixel or have similar intensity values will contribute more to the filtered result, while pixels that are in distance or have dissimilar intensity values will contribute less.



$\sigma_s = 3.0, \sigma_i = 150.0, kernel = 5$

$\sigma_s = 3.0, \sigma_i = 150.0, kernel = 11$

$\sigma_s = 3.0, \sigma_i = 100.0, kernel = 5$

$\sigma_s = 3.0, \sigma_i = 100.0, kernel = 11$