CSC2529-hw1

1010171181 Xinran Zhang

September 2023

1 Task1

1.1 Formulas

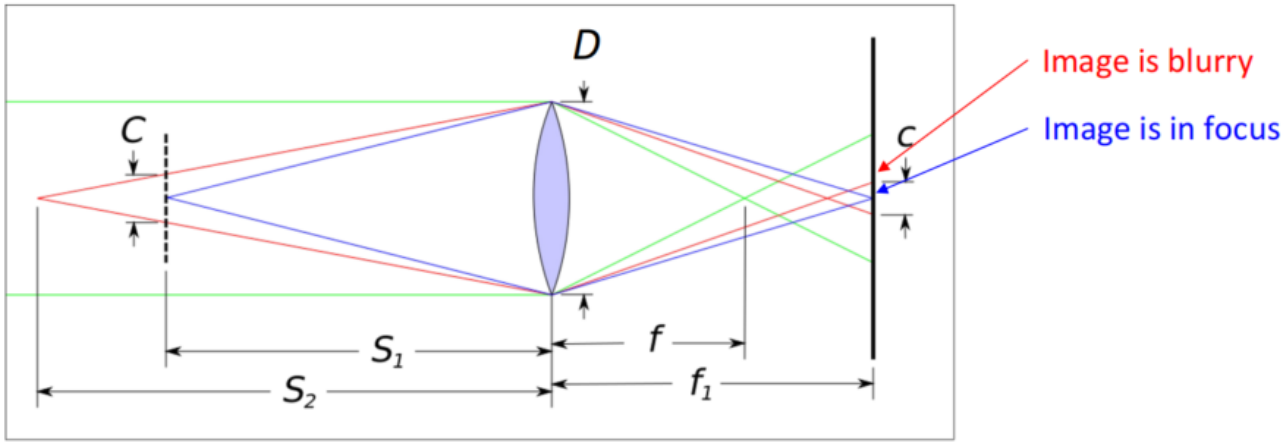

Figure 1: Optical Path Diagram

This task provides us with 'circle of confusion', which refers to the maximum diameter of the blurred area. We understand that if an object is positioned closer or farther than the focal length, the resulting image will be blurry. Therefore, to calculate the depth of field, it is necessary to determine the distance range, encompassing both the farthest and closest distances.

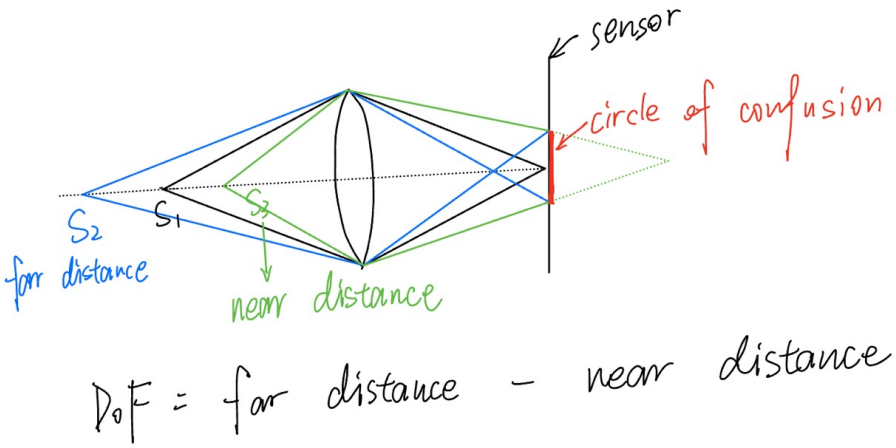

Figure 2: Depth of Field Diagram

Here are some formulas.

$$N = \frac{f}{D} \tag{1}$$

$$M = \frac{f}{S_1 - f} \tag{2}$$

$$\frac{1}{f} = \frac{1}{S_1} + \frac{1}{f_1} \tag{3}$$

$$c = M * D * \frac{|S_1 - S_2|}{S_2} = M * \frac{f}{N} * \frac{|S_1 - S_2|}{S_2} \tag{4}$$

The task provides c, D and f. Therefore, we can calculate the far distance and near distance. We use mm as the unit for all quantities. We need to convert the unit of 'c' from pixels to millimeters.

$$f = 50 \tag{5}$$

$$c = 5 * 36/5616 \tag{6}$$

In this case, we need to calculate far distance S_2 and near distance S_3 by using the formula(4). We know that $S_2 > S_1$, and $S_3 < S_1$. We can use M, D, S_1 and c to calculate these two distances. Here are the formulas.

$$\begin{aligned} M * D * \frac{(S_2 - S_1)}{S_2} &= c \\ M * D * S_2 - M * D * S_1 &= c * S_2 \\ (MD - c) * S_2 &= M * D * S_1 \\ S_2 &= \frac{M * D * S_1}{M * D - c} \end{aligned} \tag{7}$$

Similarly, we can calculate S3.

$$\begin{aligned} M * D * \frac{(S_1 - S_3)}{S_3} &= c \\ M * D * S_1 - M * D * S_3 &= c * S_3 \\ (MD + c) * S_3 &= M * D * S_1 \\ S_3 &= \frac{M * D * S_1}{M * D + c} \end{aligned} \tag{8}$$

Finally, we can calculate the DoF.

$$\begin{aligned} DoF &= fardistance - neardistance \\ &= S_2 - S_3 \end{aligned} \tag{9}$$

1.2 (a)

In the part(a), we change the aperture setting, which means we change the number N. It will subsequently impact the value of D and finally influence the DoF. Here are the results.

1. When the aperture is f/3
   (a) the near distance is 2284.71 mm
   (b) the far distance is 2760.08 mm
   (c) the DoF is 475.37 mm
2. When the aperture is f/16
   (a) the near distance is 1663.82 mm
   (b) the far distance is 5025.77 mm
   (c) the DoF is 3361.95 mm

We can find that under varying conditions, the Depth of Field (DoF) varies as well. When the aperture value, represented by N, is increased, the DoF also expands.

1.3 (b)

In this part, we define N as 3, and change the focus distances, which means we change the S_1 in the previous formulas. Because S_1 is changed, the number M is also changed. Here are the results.

1. When the focus distance is 500mm
   (a) the near distance is 491.49 mm
   (b) the far distance is 508.81 mm
   (c) the DoF is 17.31 mm
2. When the focus distance is 2000mm
   (a) the near distance is 11316.65 mm
   (b) the far distance is 85950.41 mm
   (c) the DoF is 74633.76 mm

We can find that under varying conditions, the Depth of Field (DoF) varies as well. When the focus distance, represented by S_1, is increased, the DoF also expands.

2 Task2

Here is the initial picture used in this task. We are required to apply 3 different methods to complete the demosaicing algorithm.

Figure 3: lighthouse

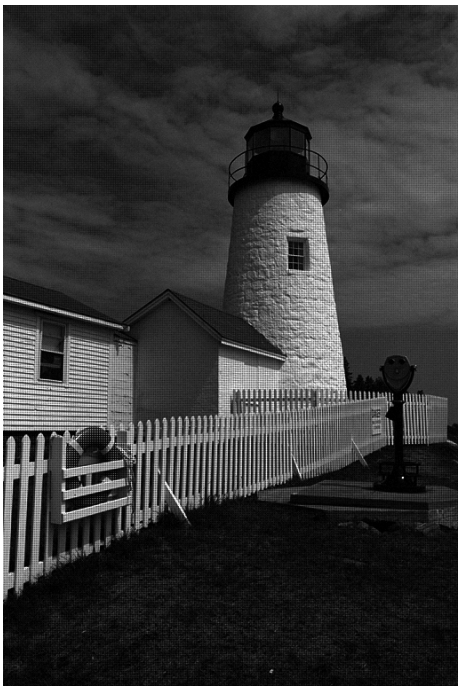

Figure 4: lighthouse RAW noisy sigma0.01

2.1 Linear Interpolation

In this section, I utilized linear interpolation to perform demosaicing, applying this technique individually to each color channel.

Initially, I divided the original picture's dataset into three color channels. Next, for the green channel, I employed np.roll() to shift the channel and subsequently averaged the shifted versions with the original green channel.

(Shift the original image in each of the four directions—up, down, left, and right—resulting in four values in previously empty areas. Then, calculated the average of these four values. Preserved the original values in the areas where values already exist.)

For the red and blue channels, I applied 'scipy.interpolate.interp2d' to interpolate.

Finally, I combined the 3 channels into an RGB image using 'np.stack([R, G, B], axis=2)'. Also, I applied a gamma correcting to convert the processed picture to a proper sRGB image. Here is the processed image.

Figure 5: Result of Linear Demosaicing

I calculated the PSNR, which is 27.1453 in this case. Compared to the light house.png, there are some color artifacts in the fence.

2.2 Median Filter

In this section, instead of applying method on RGB image, I initially converted the image into YCrCb color space. The converted dataset still has 3 layers– a luminance channel and 2 chrominance channels (Cr and Cb). Subsequently, I applied a median filter to these 2 chrominance channels and then converted back to RGB while applying the sRGB gamma. Finally, I adjusted the filter size parameters, ranging from 1 to 30, to minimize color artifacts.

Here are the results in figure 6. We can observe that a larger filter size can result in smaller color artifacts.

Subsequently, I computed the PSNR for each filter size and presented them in a line graph displayed as Figure 7. Notably, the highest PSNR value, which is 28.9745, is achieved when the filter size is set to 5.

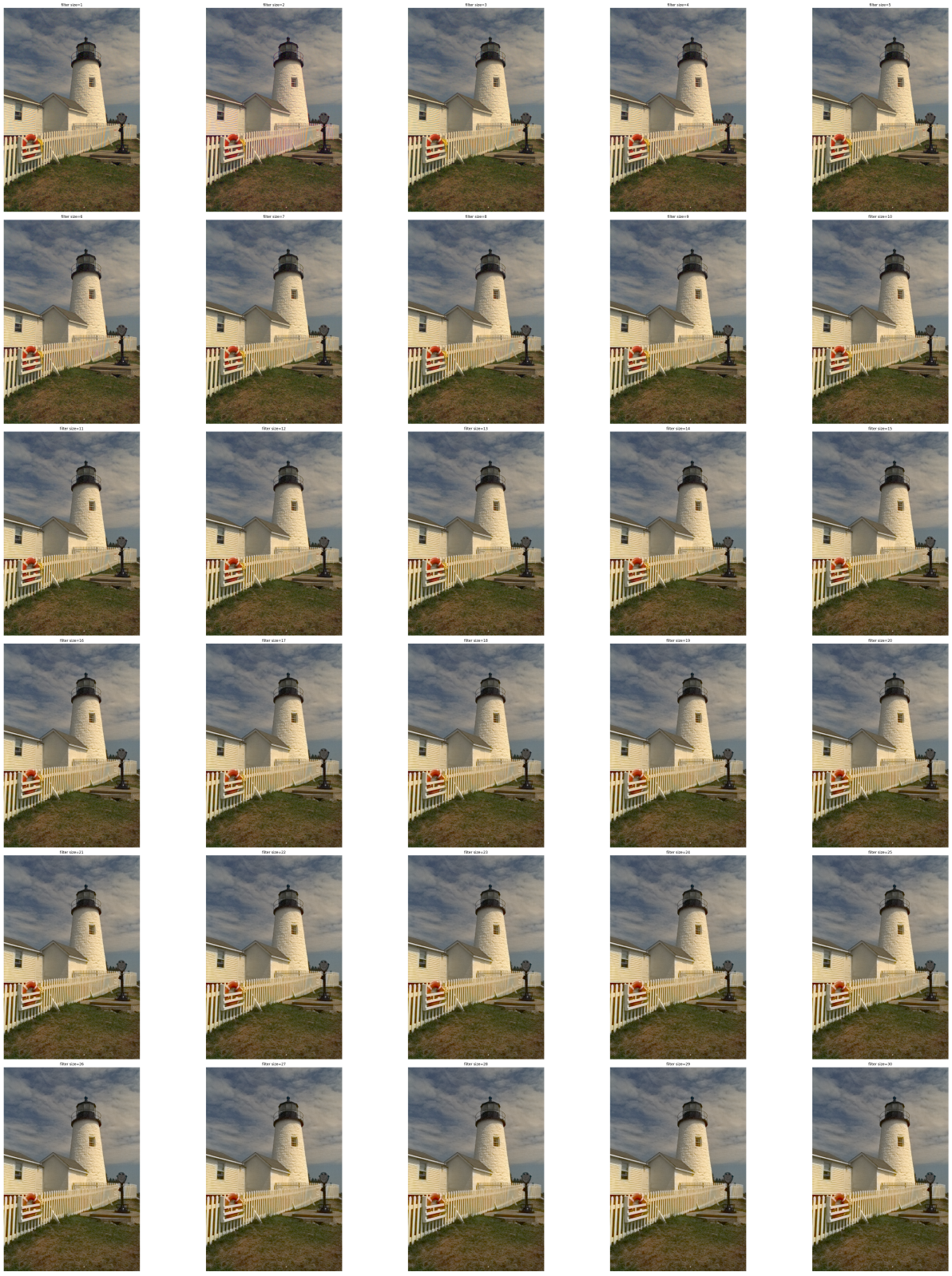

Figure 6: Images Processed with Different Filter Sizes

```
when filter size is 1 psnr for linear+smoothing is 27.145292654295076
when filter size is 2 psnr for linear+smoothing is 27.25288586389757
when filter size is 3 psnr for linear+smoothing is 28.68731434061027
when filter size is 4 psnr for linear+smoothing is 28.786256760426635
when filter size is 5 psnr for linear+smoothing is 28.974511010638537
when filter size is 6 psnr for linear+smoothing is 28.715482471412628
when filter size is 7 psnr for linear+smoothing is 28.540360922472026
when filter size is 8 psnr for linear+smoothing is 28.376058277025265
when filter size is 9 psnr for linear+smoothing is 28.2907834492671
when filter size is 10 psnr for linear+smoothing is 28.17762529381244
when filter size is 11 psnr for linear+smoothing is 28.13869129673277
when filter size is 12 psnr for linear+smoothing is 28.066034212021446
```

Figure 7: PSNR obtained by processing images with different sizes

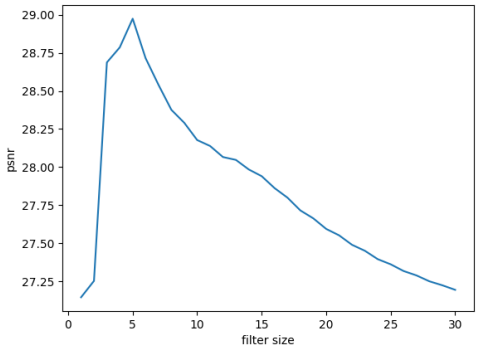

Figure 8: line graph of PSNR

2.3 High Quality Linear Interpolation

To use high quality linear interpolation, I initially devided the data set into 3 color channles and then I built 4 filters to calculate different color channels. Filter1 is used to calculate the missing values in the green layer based on the red and blue values. Filter2 is used to calculate the missing values in the red or blue layer based on the green values. Filter3 is also used to calculate the missing values in the red or blue layer based on the red and blue values. The main difference between Filter2 and Filter3 lies in the distribution of similar-colored points in their vicinity. Filter4 is used to calculate the missing values in the red or blue layer based on the blur and red values.

Next, I applied 'scipy.signal.convolve2d()' with each filter to perform filtering using the demosaicking kernels, resulting in four M*N layers. Subsequently, I extracted the necessary values from these four layers and assigned them to their respective color channels. Finally, I combined the three color channels to generate the final image.

Furthermore, I applied gamma correction to enhance the image. Here is the final processed image.

Figure 9: Optical Path Diagram

The PSNR is 28.6583

2.4 comparisons

From a qualitative standpoint, both the first and third methods exhibit relatively severe color artifacts. It is quite apparent that there are some strange colors on the fence when compared to reality in these methods. As for the second method, when the filter size is small, these color artifacts are also noticeable. However, interestingly, when the size increases, these artifacts tend to diminish.

From a quantitative standpoint, we compare the PSNR. We can observe that Method 1 has the lowest PSNR and performs the worst. When the filter size for Method 2 falls within the range of 3 to 6, its PSNR is greater than that of Method 3. Conversely, when the filter size is less than or equal to 2 or greater than or equal to 7 in Method 2, its PSNR is lower than that of Method 3. This implies that when the size is 3, 4, 5, or 6, Method 2 outperforms Method 3, whereas in all other cases, Method 3 exhibits the best performance.

	linear interpolation	Median filter (size=2)	Median filter (size=3)	Median filter (size=5)	Median filter (size=6)	Median filter (size=7)	high quality
PSNR	27.1453	28.2529	28.6873	28.9745	28.7155	28.5404	28.6583

Table 1: PSNR of different methods

3 Task3

In this task, we are required to use 4 methods to denoise the given picture. Here is the original picture with noise.

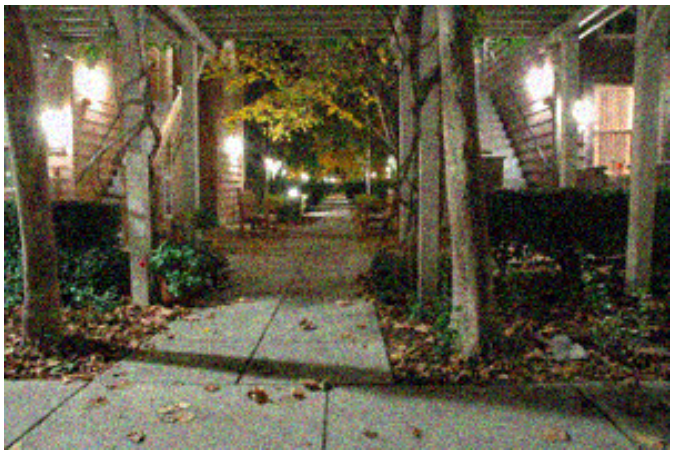

Figure 10: Initial picture in task3

3.1 Gaussian Filter Kernel

I used 'skimage.filter.gaussian' to implement a Gaussian filter kernel. Additionally, this function includes another parameter, 'channel_axis,' which provides the convenience of specifying the channel of the input image to be processed. This approach is easier and more straightforward compared to the function provided in 'fspecial.py'.

In the initial setup, the sigma parameter values were set to 1, 2, and 3, with the following results.

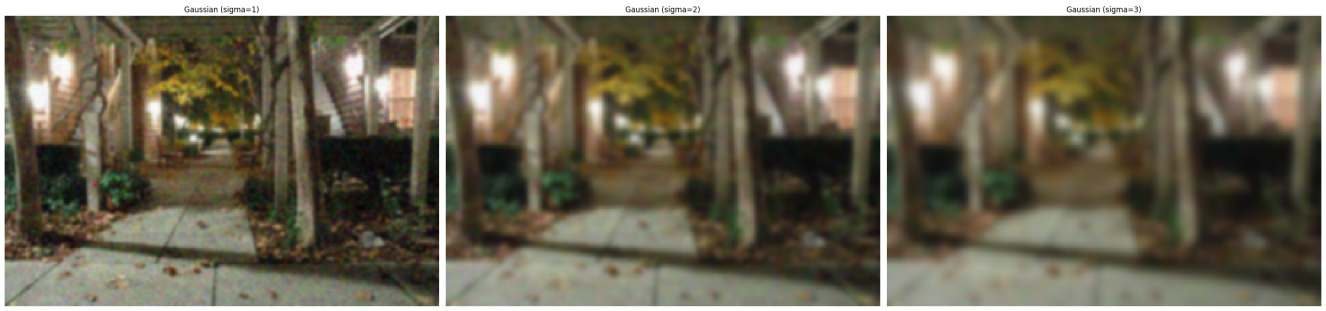

Figure 11: Processed pictures using Gaussian filter with sigma=1, 2, 3

I found that When this parameter is set to 2 or 3, the processed image is too blurry. Compared to 2 and 3, the best performance was achieved when this parameter was set to 1. Consequently, I selected 0.5, 1, and 2 as alternative parameters and reprocessed the data. The processed images are shown in the first column of Figure 13.

3.2 Median Filter

I used 'scipy.ndimage.median_filter' to realize median filter kernel. The filter size is calculated as 2*sigma+1. The initial filter size values were set to 3, 5 and 7. Here are the processed pirctures.

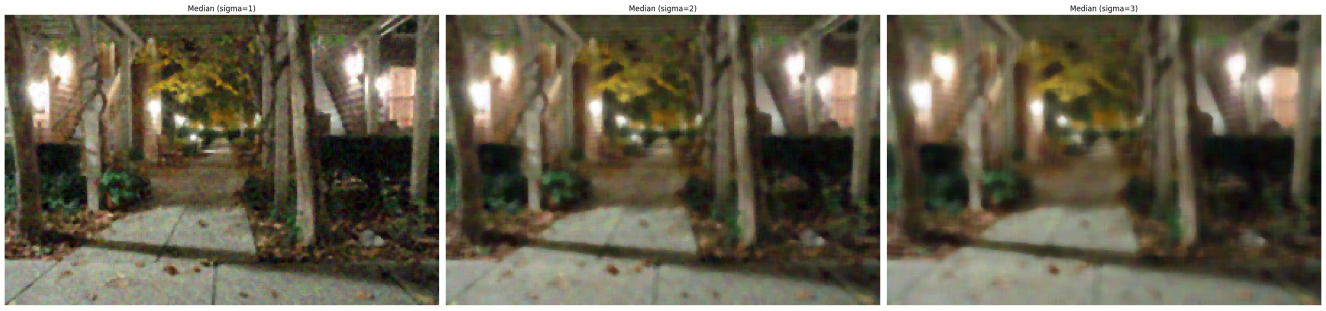

Figure 12: Processed pictures using Median filter with filter size =3, 5, 7

I found that the best performance was also achieved when the sigme was set to 1. Consequently, I chose alternative sigmas of 0.5, 1, and 2, which correspond to filter sizes of 2, 3, and 5, respectively. The processed images are displayed in the second column of Figure 13.

3.3 Bilateral Filter

I tested the same noisy image with the same 3 spatial filters with the same standard deviations – 0.5, 1 and 2. Also, I set two different values for standard deviations for the intensity σ – 0.25 and 0.5. The processed results are shown in the third and forth columns of Figure 13. The third column shows the processed pictures with intensity σ=0.25, while the forth column shows the processed pictures with intensity σ=0.5.

3.4 Non-local Means Algorithm

I realized non-local means algorithm in this part. I skipped the central pixel when comparing the similarity and used the biggest weight as the central weight - w(i,i). The processed results are shown in the fifth column of Figure 13.

3.5 Processed Pictures

As illustrated in Figure 13, each column presents images processed using a different method. The first column displays pictures processed with a Gaussian filter, the second column contains images processed with a median filter, the third and fourth columns show pictures processed with a Bilateral filter using two different intensity values for σ, and the fifth column showcases images processed using the Non-local Means Algorithm. Also, the sigma for each method are consistent: 0.5, 1, or 2.

From these processed pictures, we can observe that as the σ value increases, the noise in the pictures decreases; however, correspondingly, the images also become more blurry. In the case of the Bilateral filter, increasing the intensity σ has a similar effect, reducing noise while increasing image blur.

In principle, Gaussian filter and median filter are simpler and computationally faster. On the other hand, bilateral filter and non-local means algorithm require handling more data and are computationally slower, with non-local means being the slower of the two. What's more, the non-local means algorithm has the capability to involve information from pixels at a greater distance, which allows it to preserve more details.

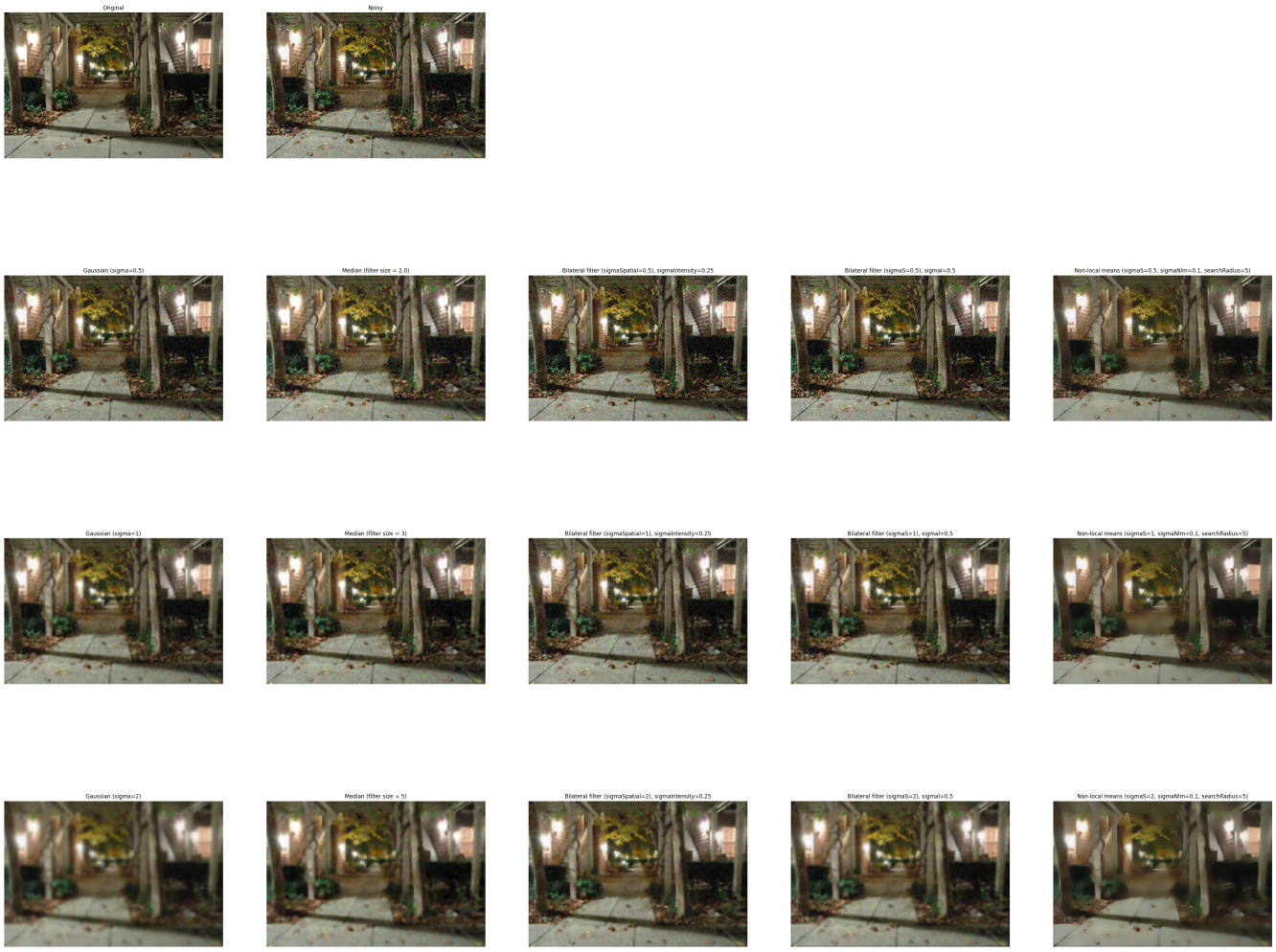

Figure 13: Optical Path Diagram