Project: A Study Of Bilateral Filtering

Student name: Yunqi Xu

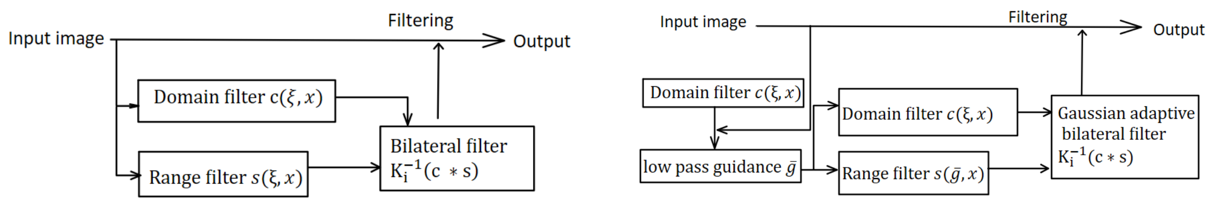
Student id: 40130514

December 9, 2022

1 Bilateral filter and Gaussian adaptive bilateral filter

Bilateral filter(BF in the following) is an edge-preserving, noise-removing, simple and non-iterative, color-suitable filtering proposed by Tomasi in 1998 [1]. The purpose of BF solves the problem of the blurred edges when filtering with a Gaussian low-pass filter, and the unstable output brought by iterative Diffusion methods.

The BF is consisted of the weighted sum of the multiplication of two gaussian-like filters, as shown in fig. 1(a), where the Domain filter $c(\xi, x) = e^{-\frac{1}{2}(\frac{d(\xi,x)}{\sigma_d})^2}$ calculate the Euclidean distance between ξ and x, and the Range filter $s(\xi, x) = e^{-\frac{1}{2}(\frac{\delta(f(\xi),f(x))}{\sigma_r})^2}$ calculate the difference between two intensity values. By using BF, a noised image could be successfully smoothed, and at the same time, edges in this image could be preserved, as indicated in [1]. The paper also mentions that BF is suitable for a color image. However, the evaluation of the paper lacks a quantitive comparison with the benchmark. Also, the smooth is not clean enough since the range kernel generated in BF is a noisy kernel [2].


Input image
Filtering
Output
Domain filter $c(\xi, x)$
Range filter $s(\xi, x)$
Bilateral filter
$K_i^{-1}(c * s)$
Input image
Filtering
Output
Domain filter $c(\xi, x)$
low pass guidance $\bar{g}$
Domain filter $c(\xi, x)$
Range filter $s(\bar{g}, x)$
Gaussian adaptive bilateral filter
$K_i^{-1}(c * s)$

(a) Processing of Bilateral filter (b) Processing Gaussian adaptive bilaterl filter

Figure 1: Comparison between Bilateral filter and Gaussian adaptive Bilater filter

For solving the problem mentioned above, the Gaussian adaptive bilateral filter(GABF in the following) which is a variation format of BF is published in 2020 [2]. The GABF can reduce the influence of the noised-contained range filter by two non-identical filters.

The same as Bf, GABF calculates the weighted sum of the multiplication of two gaussian filters, however, the range filter is generated from the input image with domain filter as shown in 1(b). So, the range filter in GABF comes from the low-pass guidance $\bar{g}$. The result in [2] indicates the success of the GABF. The author uses some popular evaluation methods assess GABF, BF, and some other variations of BF on some public datasets. As the result mentions, all of these three evaluation methods prove that the GABF can optimize input image better than others methods.

References

[1] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998, pp. 839–846.

[2] B.-H. Chen, Y.-S. Tseng, and J.-L. Yin, “Gaussian-adaptive bilateral filter,” *IEEE Signal Processing Letters*, vol. 27, pp. 1670–1674, 2020.

[3] A. Horé and D. Ziou, “Image quality metrics: Psnr vs. ssim,” in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369.

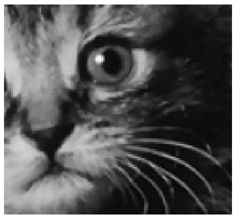
2 Experiment

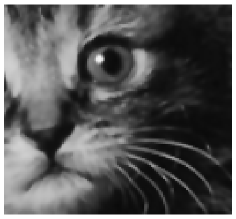
For a better understanding Bilateral filter, we will re-implement the Bilateral filter using python and present the result in this section. We will present the result on both gray and color images in Section 2.1. In section 2.2, we will use PSNR as the evaluation criteria to compare the re-implement Bilateral filter with other baseline low-pass filter methods to prove the benefit of the Bilateral filter.

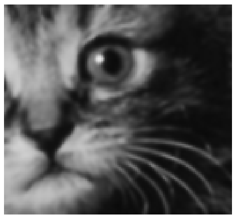
2.1 Experiment result of the Re-implement algorithm

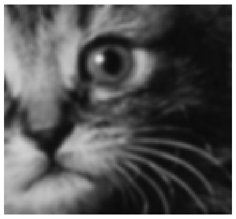
The main purpose of this section is to provide an improvement of the success of the re-implement of the Bilateral filter. During our experiment, the input images including all images come from the Bilateral filter paper [1] and some others from the internet. we will input the same parameters as the original paper suggested. In the end, we compared ours with the outputs printed on paper, and also compare with the output from the OpenCV-python official build-in function. All of this output indicates the success of our re-implement algorithm. We presented the results of the gray images in Section. 2.1.1 and colorful image in Section. 2.1.2.

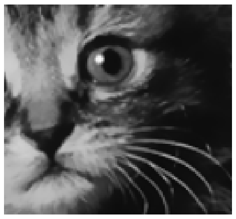
2.1.1 Experiment result of gray images

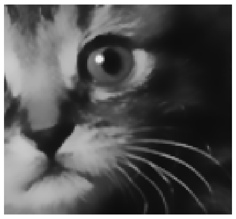

(a) $\sigma_d = 1, \sigma_r = 10$

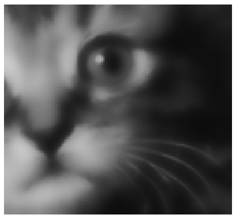

(b) $\sigma_d = 1, \sigma_r = 30$

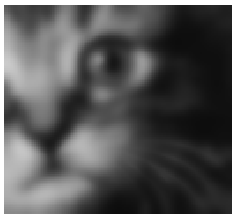

(c) $\sigma_d = 1, \sigma_r = 100$

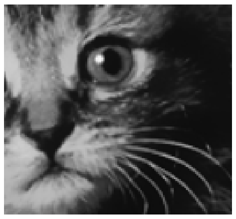

(d) $\sigma_d = 1, \sigma_r = 300$

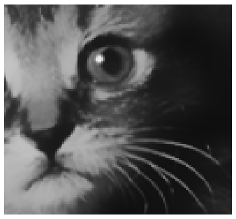

(e) $\sigma_d = 3, \sigma_r = 10$

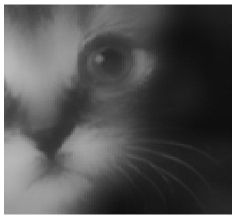

(f) $\sigma_d = 3, \sigma_r = 30$


(g) $\sigma_d = 3, \sigma_r = 100$


(h) $\sigma_d = 3, \sigma_r = 300$


(i) $\sigma_d = 10, \sigma_r = 10$


(j) $\sigma_d = 10, \sigma_r = 30$


(k) $\sigma_d = 10, \sigma_r = 100$

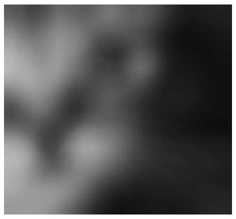
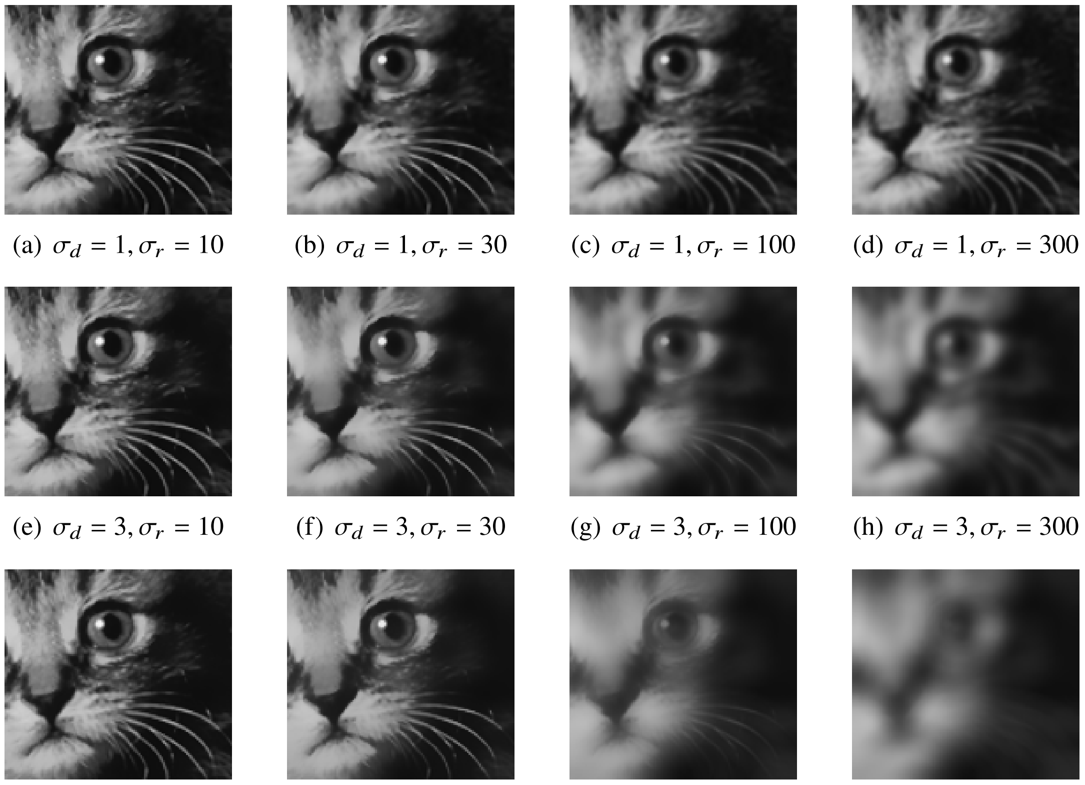

(l) $\sigma_d = 10, \sigma_r = 300$

Figure 2: A detail of cat processed with bilateral filters with various range and domain parameter values

Fig. 2 presents the output of our re-implement Bilateral filter. In our experiment, the $kernel\_size = 23$ is a fixed value through all our filtering. We choose the permutations of parameters $\sigma_d = (1, 3, 10)$ and $\sigma_r = (10, 30, 100, 300)$ forming parameter pairs. The smooth result of the re-implement Buialteral filter achieves a very similar result compared with Figure 3 from the Bilateral filter paper. Fig. 2(a) has the cleanest result, but part of the small noises remain in the image. On the other hand, Fig. 2(l) is the most ambiguous one in these outputs. It only remains a faint profile of the cat.

As a comparison, we also input the image with the same parameter pairs into the built-in OpenCV-python function to explore whether it will have the same output as our code. Fig. 3 proves the results are similar.



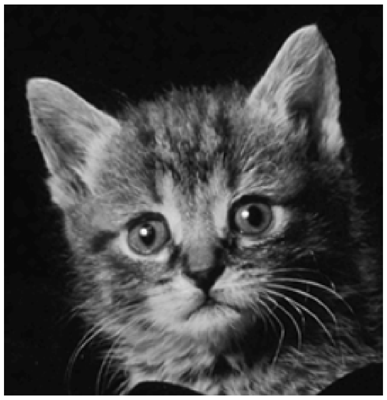
(a) $\sigma_d = 1, \sigma_r = 10$ (b) $\sigma_d = 1, \sigma_r = 30$ (c) $\sigma_d = 1, \sigma_r = 100$ (d) $\sigma_d = 1, \sigma_r = 300$

(e) $\sigma_d = 3, \sigma_r = 10$ (f) $\sigma_d = 3, \sigma_r = 30$ (g) $\sigma_d = 3, \sigma_r = 100$ (h) $\sigma_d = 3, \sigma_r = 300$

(i) $\sigma_d = 10, \sigma_r = 10$ (j) $\sigma_d = 10, \sigma_r = 30$ (k) $\sigma_d = 10, \sigma_r = 100$ (l) $\sigma_d = 10, \sigma_r = 300$

Figure 3: A detail figure with bilateral filters with various range and domain parameter values by Opencv python

Fig. 4 presents the filtering result between the original input image and the output with parameter $\sigma_d = 3$ and $\sigma_r = 30$. The output is very similar to Figure 5 in the Bilateral filter paper. Also, The property of edge-preserving is shown in these images, the Kitten's whiskers are preserved after filtering.


(a) $\sigma_d = 1, \sigma_r = 10$

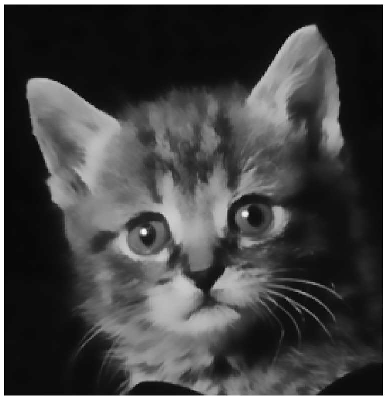
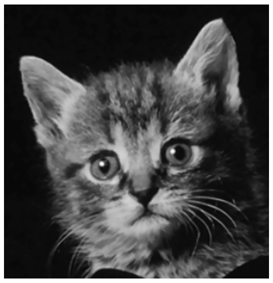
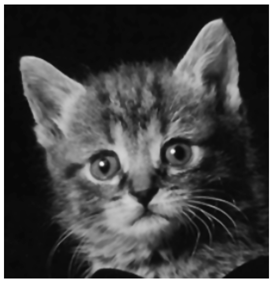
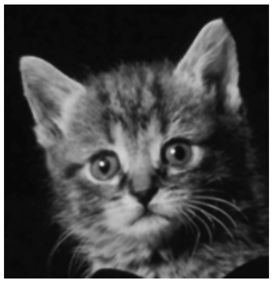
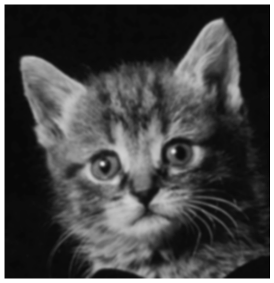

(b) $\sigma_d = 3, \sigma_r = 30$

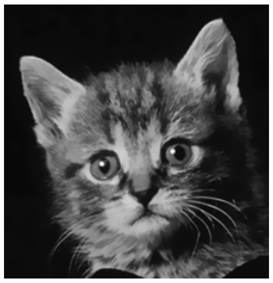
Figure 4: Output of cat

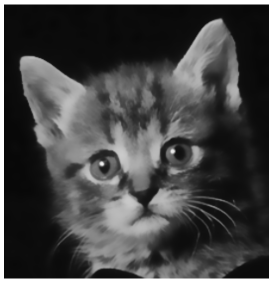

(a) $\sigma_d = 1, \sigma_r = 10$

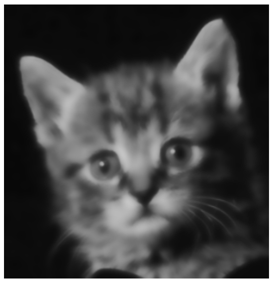

(b) $\sigma_d = 1, \sigma_r = 30$

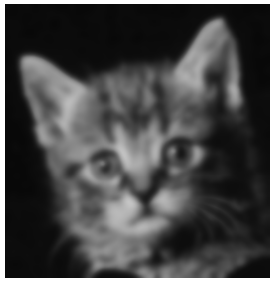

(c) $\sigma_d = 1, \sigma_r = 100$

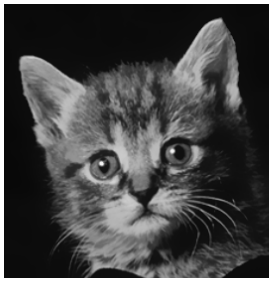

(d) $\sigma_d = 1, \sigma_r = 300$

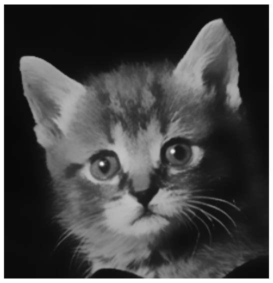

(e) $\sigma_d = 3, \sigma_r = 10$

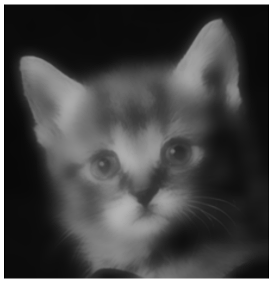

(f) $\sigma_d = 3, \sigma_r = 30$


(g) $\sigma_d = 3, \sigma_r = 100$


(h) $\sigma_d = 3, \sigma_r = 300$


(i) $\sigma_d = 10, \sigma_r = 10$


(j) $\sigma_d = 10, \sigma_r = 30$


(k) $\sigma_d = 10, \sigma_r = 100$

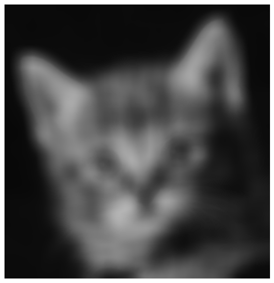
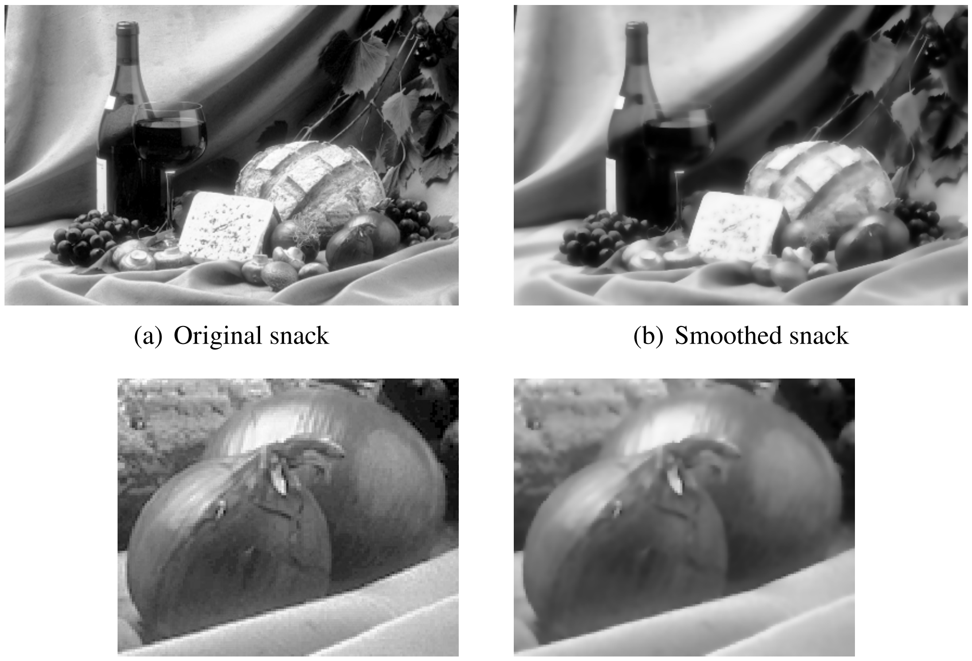

(l) $\sigma_d = 10, \sigma_r = 300$

Figure 5: A detail figure with bilateral filters with various range and domain parameter values by re-implement code of cat

Fig. 5 are the whole output of the input image 4(a) with the same parameter pairs as before.


(a) Original snack (b) Smoothed snack

(c) Original Onion (d) Smoothed Onion

Figure 6: The Bilater filtering result of snacks and the Onion detail

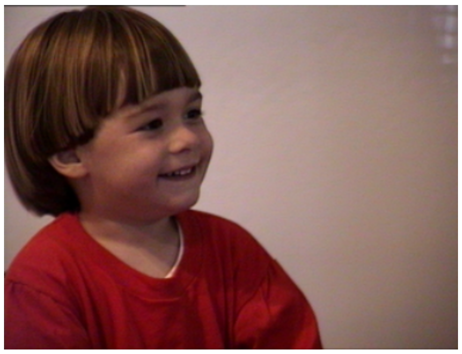
The Fig. 6 presents the result of Bilateral filter (Fig. 6(b), 6(d)) and their original images Fig. 6(a) and 6(c). The choosed parameter pair ($\sigma_d = 3$ and $\sigma_r = 50$) is the same as paper mentioned. As shown in Fig. 6(b), the salt and pepper noises can be removed. Meanwhile, the edges can be preserved. Fig. 6(d) is the detail of Fig. 6(b).

2.1.2 Experiment result of color images

The bilateral filter is not only suitable for filtering the gray image, but also achieving success for filtering color images. In the paper, the author mentioned that transforming color space from RGB to CIE-lab color space can solve the color distorted problem directly dealing with RGB color space.

Fig. 7 has presented the result of filtering on color images with CIE-lab space. Fig. 7(b) exhibits a similar result as printed on the paper.

Fig. 8 also presents some other noised images from the internet which can be smoothed and edge-keeping with Bilateral filtering. In these images, the first row (Fig. 8(a), 8(b), 8(c)) are original images, and the second row (Fig. 8(d), 8(e) and 8(f)) are the smoothed result.


(a) Child

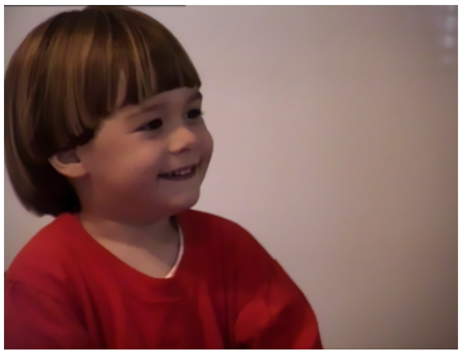
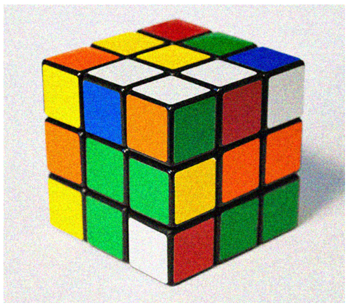
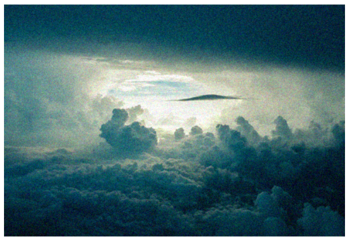
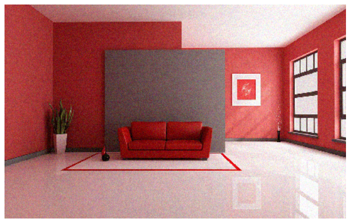
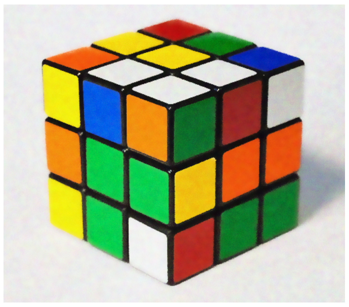

(b) Smoothed Child

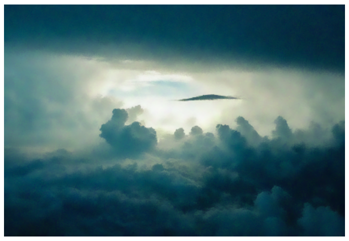
Figure 7: Output of child image


(a) Rubik's cube


(b) Sky


(c) Home


(d) Smoothed Rubik's cube


(e) Smoothed sky

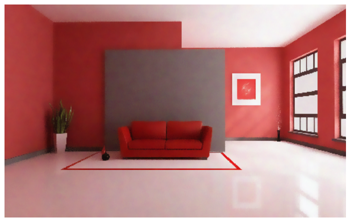

(f) Smoothed home

Figure 8: Outputs of other color images

2.2 Compare with other baseline algorithm

In this section, we compared the results of the Bilateral filter with other baseline algorithms to present the advantages of our code. Before the Bilateral filter, the gaussian filter is one of the useful methods which can smooth a noised image. Other low-pass filters, such as the Mean filter and Median filter will also be considered in our experiment. Since all of these filters are classic filtering methods in terms of image processing.

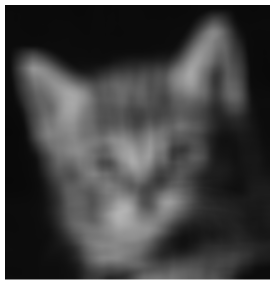
In the Section. 2.1, our re-implement has proved the success of the Bilateral filtering method for gray and color images. But only from the versioning aspect is not enough, sometimes, the human version may produce bias.

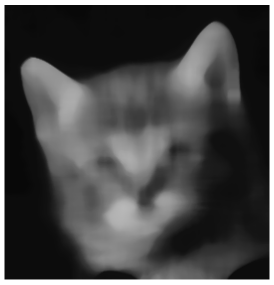
So, in our experiment, we use the quantitive method to compare our Bilateral filter output with different baseline low-pass filters with PNSR. The equation of PNSR has been shown in Eq. 1.

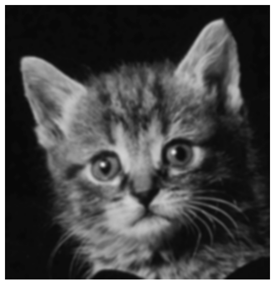
$$PSNR = 10 \log 10(\frac{I_{Max}^2}{MSE}) \tag{1}$$

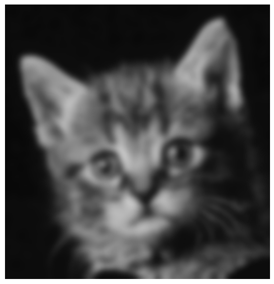
$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N} \tag{2}$$

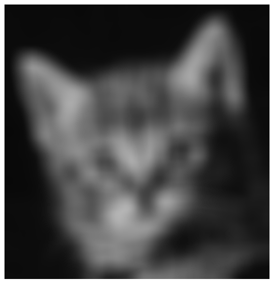
Where the I_{Max} is the maximum fluctuation in the input images. In our case, the 8-bit integer input image should have $I_{Max} = 255$. And the I_1 and I_2 are the two images between the noisy and the filtered one.


(a) $Mean_{psnr} = 19.89$


(b) $Median_{psnr} = 20.66$


(c) Gaussian $\sigma = 1$


(d) Gaussian $\sigma = 3$


(e) Gaussian $\sigma = 10$

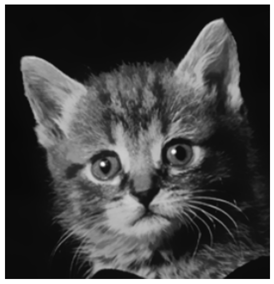

(f) Bilateral filter output

Figure 9: output compared with the baseline low pass filters in gray scale

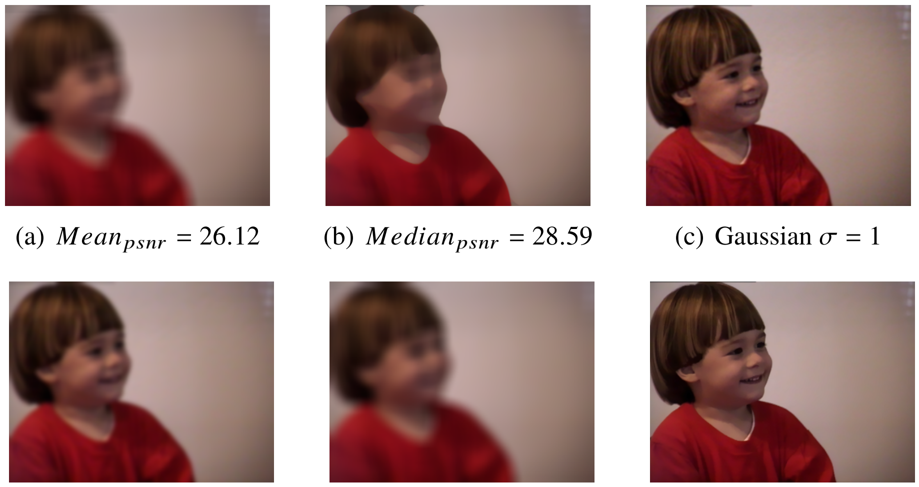
Fig. 9 has presented the result between three baseline filters and the Bilateral filter. It is obvious that Fig. 9(a) and Fig. 9(b) get bad results, due to the reason that all the details and texture information are lost. The only remaining parts are the contour of the cat in each image. The result of Gaussian filter, as shown in Fig. 9(c), 9(d), 9(e), indicates that the smaller σ of gaussian filter achieves better result. However, with the increase of the $sigma$ values, the result of the gaussian filter gets fuzzy. The Fig. 9(c) and Fig. 9(f) prove the statement in [1]. Gaussian blur will also smooth the edges, in Fig. 9(c), the whiskers of the cat also become fuzzy, but still maintains clear in Fig. 9(f).

Table. 1 presents the PSNR result. As the result in the table, gaussian blur have some lower scores when $\sigma = sigma_d$ in Bilateral filtering. The PSNR score of Gaussian close the score of the Bilateral filter when $\sigma_r = 300$. But for each $sigma_d = (1, 3, 10)$, the most ambiguous result is obtained when $sigma_r = 300$.

Method	Kernel_size	sigma_s	Sigma_r	PSNR
Bilateral Filter	23	1	10	42.74
			30	36.26
			100	32.59
			300	31.70
Gaussian filter(σ = 1)	23			31.59
Bilateral Filter	23	3	10	40.05
			30	31.37
			100	25.90
			300	24.39
Gaussian filter(σ = 3)	23			24.13
Bilateral Filteral	23	10	10	39.59
			30	29.74
			100	22.57
			300	20.69
Gaussian filter(σ = 10)	23			20.41

Table 1: The PSNR output for gray image

The same situation also happened filtering color images. Fig. 10 indicates that the Mean and Median blur can only maintain a contour of the object and the Gaussian filter with $\sigma = 1$ achieves the clearest output in all the Gaussian filters. It is apparent that the Bilateral filter has the highest result as shown in Table. 2.


(a) $Mean_{psnr} = 26.12$ (b) $Median_{psnr} = 28.59$ (c) Gaussian $\sigma = 1$

(d) Gaussian $\sigma = 3$ (e) Gaussian $\sigma = 10$ (f) Bilateral filter

Figure 10: output compared with the baseline low pass filters in color iamges

Method	Kernel_size	sigma_s	Sigma_r	PSNR
Bilateral Filter	23	1	10	42.91
			30	41.22
			100	39.42
			300	38.96
Gaussian filter($\sigma = 1$)	23			39.01
Bilateral Filter	23	3	10	39.02
			30	35.92
			100	32.35
			300	31.19
Gaussian filter($\sigma = 3$)	23			31.11
Bilateral Filteral	23	10	10	39.27
			30	33.18
			100	28.46
			300	26.86
Gaussian filter($\sigma = 10$)	23			26.73

Table 2: The PSNR output for color image

3 Conclusion

In a conclusion, in this report, we successfully re-implement the Bilateral filter based on the guidance of the content mentioned in [2]. We have proved that Bilateral filter can optimize better compared with other low-pass filters. Especially it reduce the negative impact on the blured edges by Gaussian filter. We also explored the quality of outputs between Bilateral filtering and other low-pass filters. By presentt the reuslt of this, we quantitively analyzed the performance of Bilateral filter by evaluating PSNR score between difference filters.

However, there are some drawbacks to our re-implement. Firstly, the Bilateral filter is a brute-force loop calculating algorithm, as it searches all the pixels from an input image. Without any search optimization of our re-implement code, the calculation will become slower when the input image size becomes large or the kernel size increase. Secondly, the evaluation method we chose is not the best one for evaluating blured pictures PSNR values sometimes do not match the visual quality perceived by the human eye, such as the PSNR does not represent the visual perceptual properties of the human eye well [3]. Also, PSNR is not sensitive to structural changes when blurring occurs. Based on this, PSNR has some limitations, it may not correctly reflect the result in some cases.

Not only the Bilateral filter,there are also many knowledges we take advantages of class. The first is PSNR, as mentioned before, we use this as an evaluation indicator to quantitive evaluate our re-implement output and other baseline low-pass filters. Also, The low-pass filters we choose, such as mean, median, and gaussian are all introduced in class. especially, we further explore the properties of the Gaussian filter and understand its advantage and drawbacks respectively. Besides, during the re-implement process, we also utilize the padding method explained in the class. The zero-padding is a easy method but not the optimal choice, so, in the end, we choose the reflection padding as our pre-processing.

There are also some difficulties during our implementation process. The first is the omission parameters in the original paper. The original paper lacked information about the chosen of kernel size. Also, since this article was published 20 years ago, finding the images used by the author at that time was a very difficult task. The resolution ratios of these images from the internet or cut from paper are different. Different resolution ratios not only influence the output result, also impact the choice of experiment parameters. These difficulties spent us a huge amount of time to figure out the solution.