COMP 6771 Image Processing: Project

Student name: Yunqi Xu

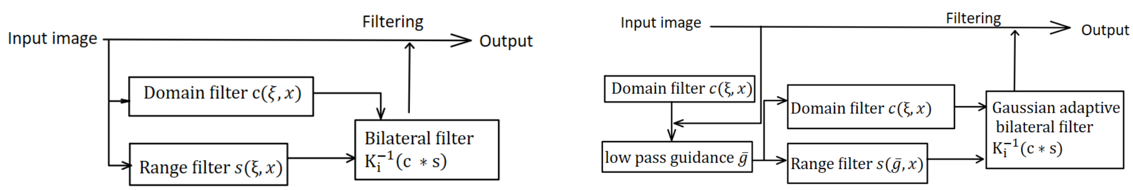
Student id: 40130514

December 8, 2022

1 Introduction

1.1 Bilateral filter and Gaussian adaptive bilateral filter

Bilateral filter was proposed by Tomasi in 1998 [1], Which is edges keeping, noises removing, simple and non-iterative,multi-channels suitable method, consisted by two filters, Geometric closeness(domain filter $c(\xi, x)$ and photometric similarity(range filter $s(\xi, x)$).



(a) Processing of Bilateral filter

(b) Processing Gaussian adaptive bilaterl filter

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

The process of utilizing Bilater filter denoising an image is presented in Fig. 1(a), where the Domain filter $c(\xi, x) = e^{-\frac{1}{2}(\frac{d(\xi,x)}{\sigma_d})^2}$, where, $d(\xi, x)$ is the Euclidean distance between ξ and x. The Range filter $s(\xi, x) = e^{-\frac{1}{2}(\frac{\delta(f(\xi),f(x))}{\sigma_r})^2}$, where, $\delta(\phi, f)$ is a suitable measure of distance between the two intensity value ϕ and f. Also the K_i^{-1} denotes a normalizing factor. The overall processing could be descrbed as using the weighted sum of multiple of two generated gaussian-like kernel filtering an input image.

The Bilateral filter can significantly solve the slow spatial variantion fails at edges, which usually occurs with tranditional Gaussian filters, and the complexity of many iteration that brought by Diffusion method. The result in the Bilateral filter [1] indicates that it is suitable from both gray and color images. However, in the evluation of the paper, the lack of quantitive comparison between bilateral filter and other baseline method make it difficult to generate though with readers.

On the other hand, another improvement version of Bilateral filter, Gaussian adaptive bilaterl filter, purposed by Chen [2] in 2020, which using two nonindentical filters solving the inherent problem occured in Gaussian range filter when facing a noise filtering input and its effect on edge-preserving image smoothing operation.

The process of Gaussian adaptive bilateral filter is in Fig. 1(b) The same as the processof Bilateral filter, the GABF use two filters, domain filter and range filter, however, the compared with BF in Fig. 1(a), the range filter is generated with a pre-calculation low pass guidence $\bar{g}$ with Gaussian domain filter. So the Range filter will calculate the intensity difference between input image and $\bar{g}$.

The result in [2], indicates the succseeful of the GABF, and also the auther use quantivative analysis such as the SSIM, PSNR and GMSD to evaluate its output, all of the output is achieve advantages compared with BF.

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 Details of Bilateral Filter

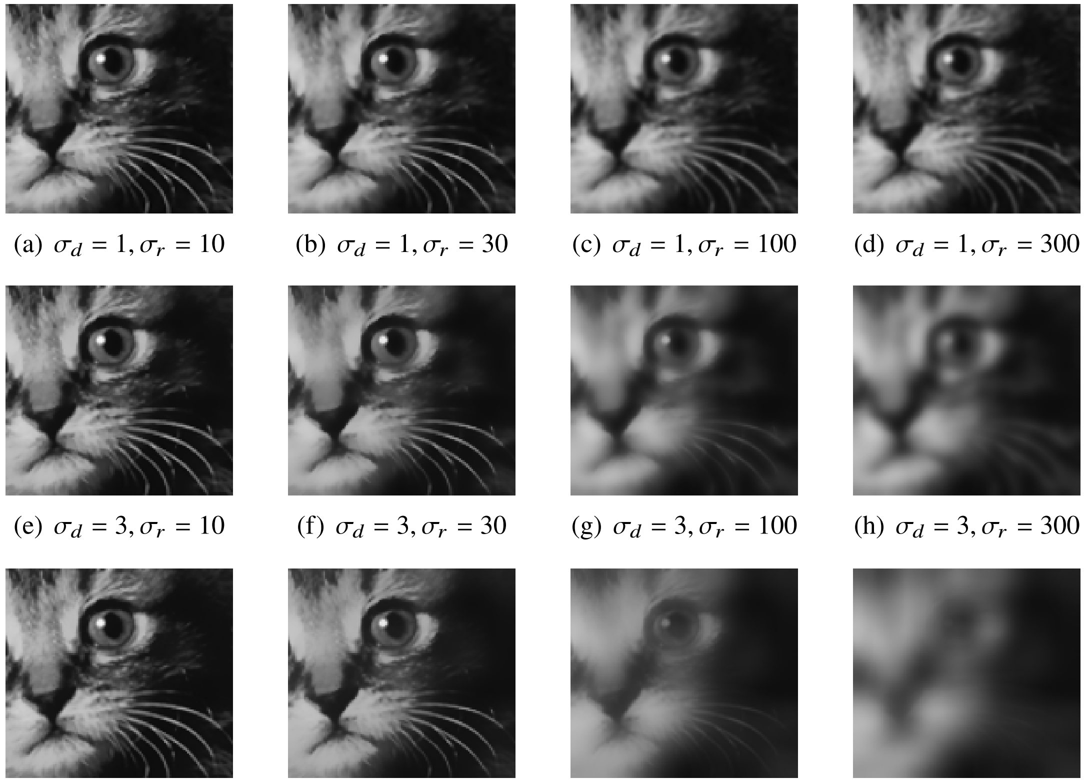
This section contains three main parts. we will firstly present the result of our re-implement method, and then compared the bilater filter with other baseline algorithm in terms of other low pass filters that usually blur the image but also blur edges. Finally, we will discuss about the Bilateral filter from both advantages and disadvantages and the difficulties during the implement process.

2.1 Result of the Re-implement algorithm

The main purpose for this section is providing an provement of the successful re-implement of the Bilateral filter. During our experiment, we use the images from the Bilateral filter paper [1], inputted the same parameters that the paper utilized, and compared our result with the oroginal paper and also the build-in function with Opencv-python offcial to prove the successfully re-implement of our code.

2.1.1 Test result on gray images

Firstly, we present our result compared with the result indicated in the paper, with the same inputted parameters and the same pattern. In the papre, the author use the parameters $\sigma_d = (1, 3, 10)$ and $\sigma_r = (10, 30, 100, 300)$ correspond to each consisting the parameter set. We also use the $kernel\_size = 25$. The Fig. 2 presents the output of our re-implement Bilateral filter. In this Figure, we can find the blur trend are similar with Figure 3. from the Bilteral filter paper. The Fig. 2(a) has the most clean result, but part of small noises still could be find in it. On the other hand, the Fig. 2(l) is the most blur one in these outputs, it only remain a blurry profile of the 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 2: A detail figure with bilateral filters with various range and domain parameter values by implement code

As the comparison, we also make an experiment of input the same parameter pairs in to the build-in opencv-python funtion to explore whether it will have the same output as our code. Fig. 3 prove the results are samilar.

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

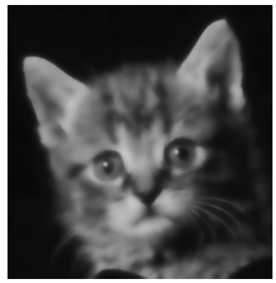
(b) $\sigma_d = 1, \sigma_r = 30$

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

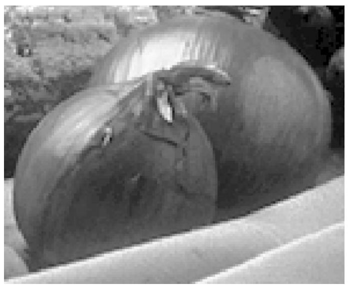
Figure 4: A detail figure with bilateral filters with various range and domain parameter values by re-implement code of cat

Fig. 4 are some other outputs use the cat image from paper and the same parameter sets. The same as the figures from the paper, some detils, such as the Kitten's whiskers, also can be remained after filtering.

The Fig. 5 are some other outputs use the figures from Bilateral filter paper. The Fig. 5(a), 5(c) are the oroginal images. As present in Fig. 5(b), the salt and pepper noise can be removed, at the mean time, the edge information can be keeped as shown in Fig. 5(d). In this experiment, we use same parameters which the paper used. $\sigma_d = 3$ and $\sigma_r = 50$.

(a) Original snack

(b) Smoothed snack



(c) Original Onion

(d) Smoothed Onion

Figure 5: The Bilater filtering result of snacks and the Onion detail

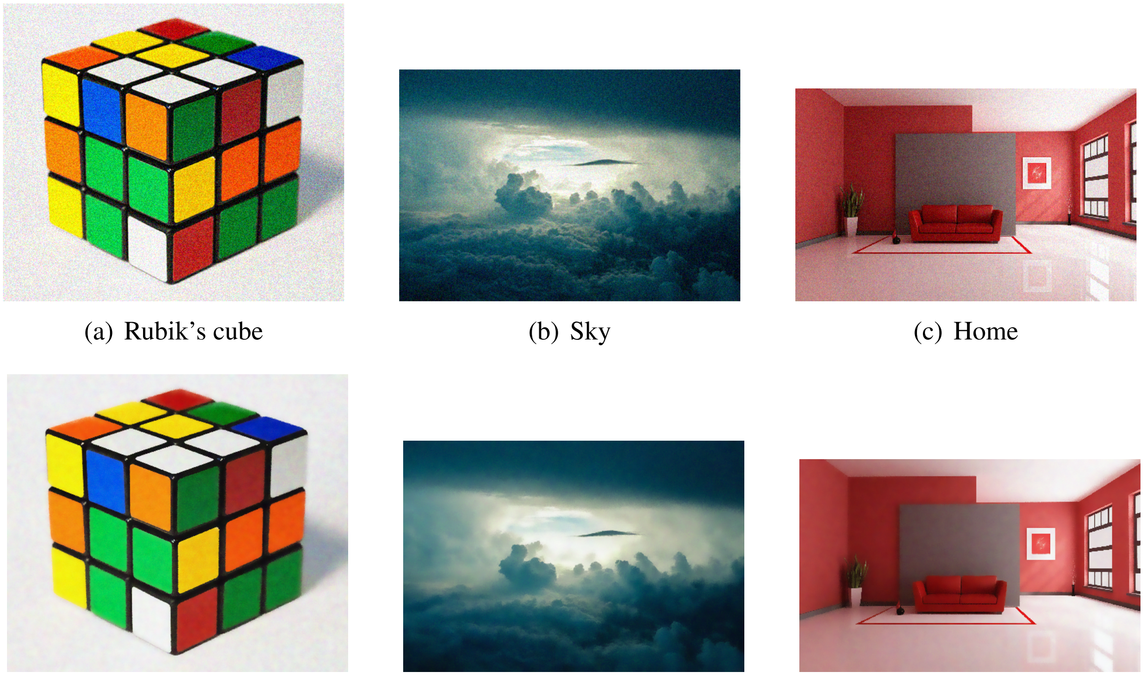
2.1.2 Test result on color images

The bilateral filter is not only suitable for filtering gray image, but also achieving successful at filtering color images. Fig. 7 has presents the result of filtering on color images. In these images, the left half (Fig. 6(a), 7(a), 7(b), 7(c))are original imput images. The right half(Fig. 6(b), 7(d), 7(e) and 7(f)) are the result of those color images with Bilater filtering. Compared with filtering gray images, filtering color images need to convert 3-channels RGB images into the CIE-lab space, and then generate two filters based on geometric and photometric information that an image provided. Noises on the original input images could be removed.

(a) Child

(b) Smoothed Child

Figure 6: Output of child image



(a) Rubik's cube (b) Sky (c) Home

(d) Smoothed Rubik's cube (e) Smoothed sky (f) Smoothed home

Figure 7: Outputs of other color images

2.2 Compare with other baseline algorithm

In this section, we compared the filter result of Bilateral filter with other baseline algorithm to present the advantages of our code. The baseline low-pass filters we used are mean filtering, median filtering and gaussian filtering. All of this filters are classic filtering in terms of image processing. The Section 2.1 not only prove the result of the re-implement code, but also indicate that the general applicability of Bilateral filter. Our next experiment, we not only compared the output result passed by different filters from human version level, but also calculate the PNSR [3] from mathmatical level. The equation of PNSR has been shwo in Eq. 1:

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

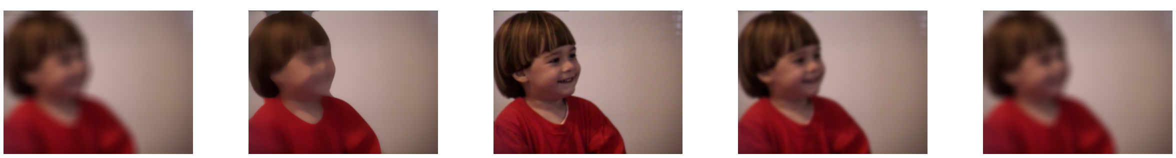
$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(M,n)]^2}{M * N} \tag{2}$$

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

(a) $Mean_{psnr}$ = 19.89 (b) $Median_{psnr}$ = 20.66 (c) Gaussian(1) (d) Gaussian(3) (e) Gaussian(10)

Figure 8: output compared with the baseline low pass filters

The Fig. 9 has presented the result between three baseline filters and the Bilateral filter. Is is obvious that the the Fig. 9(a) and Fig. 9(b) get a very bad result, since all the details and texture inforamtions are lose. The only remaining parts are the contour in each image. The result of Gaussian filter, Fig. 9(e) can achieve a very closely result compared with Fig. **??**. However, as we mentioned in Section. 2.1.1, the kitten's whiskers are clean in Fig. **??**, they are blured in Fig. 9(e). For further prove our statement, we calculate the PSNR between each output image with filter and the original noisy image. The Table. **??** presents the PSNR result. As the result in the table, the PSNR scores of Median filter and Mean filter are too low to use, but the PSNR score of Gaussian filter can achieve a considerablely compititive result as 31.59, which is higher than using Bilateral filter with $\sigma_s = 3$ and $\sigma_d = 100$ or $\sigma_d = 300$ and $\sigma_s = 10$ and $\sigma_d = 30, 100, 300$.



(a) $Mean_{psnr}$ = 26.12 (b) $Median_{psnr}$ = 28.59 (c) Gaussian(1) (d) Gaussian(3) (e) Gaussian(10)

Figure 9: output compared with the baseline low pass filters

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

Table 1: The PSNR output for gray image

The Tabel. **??** presents the PSNR reult compared with Bilater filter and other baseline filters.

As we mentioned in Section 2, the tranditional low pass filter, especially the Gaussian filter, compute the weighted average of pixel values in the neighborhood and intuition that pixels from one image have a slow spatial variation. But this assumption fails when filtering edges.

The successfully of Bilateral filter also be proved in filtering color images.

Method	Kernel_size	sigma_s	Sigma_r	PSNR
Bilateral Filter	23	1	10	42.91
			30	41.22
			100	39.42
			300	38.96
Bilateral Filter	23	3	10	39.02
			30	35.92
			100	32.35
			300	31.19
Bilateral Filteral	23	10	10	39.27
			30	33.18
			100	28.46
			300	26.86
Gaussian filter($\sigma = 1$)	25			39.01
Gaussian filter($\sigma = 3$)	25			31.11
Gaussian filter($\sigma = 10$)	25			26.73

Table 2: The PSNR output for color image

3 Conclusion

In this section, we will provide the pros and cons of the methods. The methods we used in this report that comes from the lecture. finally, we will provide some brief discription about the difficulties we faced.

3.1 Pros and Cons

Pros: In our re-implement, we prove the success of Bilateral filter in edge preserving and denoising. We also quantitive compared the Bilateral filter with other baseline denoising filtering in terms of PSNR.

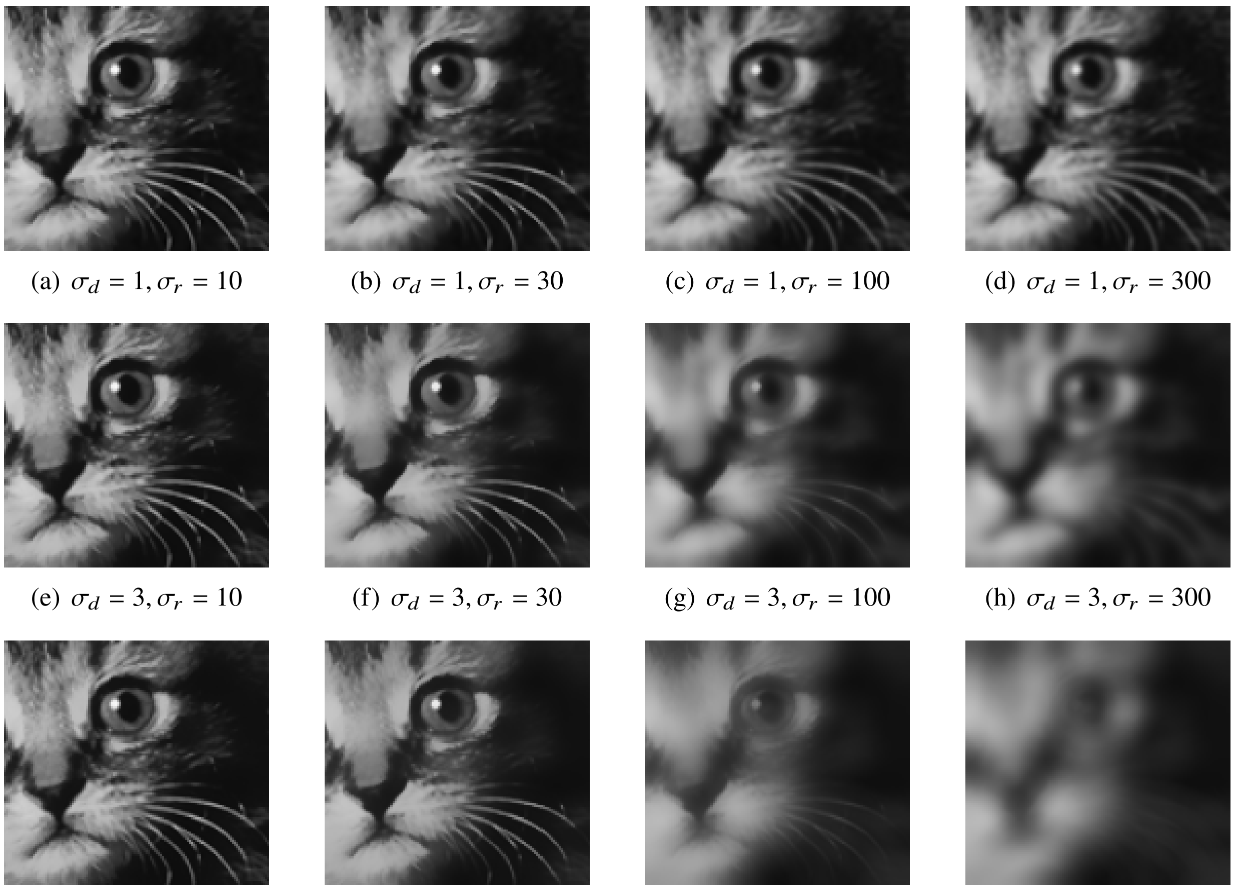
Cons: When Bilateral neighborhood size gets larger, the Bilateral filtering is slow. Since we use loop in the filtering processing to filter the input image. The larger the filter size is, the slower the algorithm will be excused. Compared with the build-in opencv-python Bilater method, our re-implement comsumsed more time especially when dealing with color and large size image with larger kernel size. On the other hand, the drawbacks of our evaluation method still insufficient. Because sometimes the PSNR may not reflect the correct subjective quality of the image. 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. Also, PSNR is not sensitive with structural changes when blurring occur. PSNR has some limination, it may not correct reflect the result in some case.

3.2 Method used

The methods we used in this paper, including padding before filtering. During our experiment, the reflect padding can provide a better result, so we keep the refect padding as the final version of our re-implement. During the comparison process, we use PSNR as the evaluation method. Also, the comparison low-pass fitler we choose are some classic blur filters, mean, median and gaussian filters.

3.3 difficulties

The first difficulties is implement the σ_r in the color image. Since all the information indicates how to calculate the σ_r in 3-channels are separate inside the paper. The second is the transform of CIE-lab space. Since the CIE-lab has different value comapred with RBG color space, the order of pre-processing images before filter become important if we want get a correct output.



(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