DIP Team01 Final Report

Group 1 B10705002 吳亞宸 B10705005 陳思如

I. Division of the work of each team member

“Fast bright-pass bilateral filtering for low-light enhancement” - 陳思如
“Fast Adaptive Bilateral Filtering of Color Images” - 吳亞宸

II. Paper Title

Combining bilateral filtering on all kinds of image enhancement at once

Ideas based on these two papers:
S. Ghosh and K. N. Chaudhury, “Fast bright-pass bilateral filtering for low-light enhancement”, Proc. IEEE International Conference on Image Processing (ICIP), pp. 205-209, Taipei, Taiwan, 2019
R. G. Gavaskar and K. N. Chaudhury, "Fast Adaptive Bilateral Filtering of Color Images," 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019, doi: 10.1109/ICIP.2019.8802987

III. Motivation

People oftenly capture images and find out that the image is not bright enough to detail out the objects afterwards. However, we may miss the opportunity to get to the place or timing again to have the exact same photo. Also, the image processing techniques have improved in recent years. Many effective procedures are introduced and implemented in both grayscale and color images. Based on these two reasons, using a one kind algorithm or method that is applicable to all different kinds of images and a reasonable low cost is the user’s goal. We want to implement an automatic enhancement and short processing time techniques to help the user to enhance the input image.

Then for the reason why we use these two algorithms,
Fast Bright-Pass Bilateral Filtering for low-light enhancement:

1. aim on solving low-light images
2. retinex model can estimate the illumination and reflectance that will generally apply on all image conditions
3. improved the brute force approach with Fourier approximation to accelerate filtering to reduce the intensive computation and long processing time

Fast Adaptive Bilateral Filtering for color images:

1. adapt the width of the range kernel at each pixel can boost the enhancement capacity of the filter which result in great performance
2. well developed and fast approach on gray images

3. approximate the histogram by uniform distribution to apply on real-time applications on color images

These two algorithms directly meet our needs in the general performance and relatively short processing time compared to other approaches.

IV. Problem Definition

We want to use the fast bilateral filtering algorithm to enhance the image brightness and details simultaneously. Fast bright-pass bilateral filtering has been introduced to solve the low-light image problem in effective and reasonable time. Fast adaptive bilateral filtering then has been implemented in image sharpening to highlight the details. These two techniques solve the time-consuming problems in other image processing algorithms. So, we want to combine these two methods to enhance the image with brightness and detail at the same time. The users can expect a desirable result just by one step instead of trying on these two algorithms separately. Also, the enhancement params will be set automatically so that the user doesn't need any manual modification on it but the user still gets desired output.

V. Algorithm

01. Fast Bright-Pass Bilateral Filtering for low-light enhancement

a. Convert to HSV and extract the V channel to modify illumination
b. Iterate with different Fourier order to find the best tone-mapping parameter with Fourier cosine expansion and sigmoid function
c. Compute channel with Fourier approximation on gaussian filter
d. Get the ration between original and compressed channel
e. Compute the gamma correction on compressed channel
f. Reconstruct the V channel by the ratio and gamma correction
g. Convert back to RGB to get the enhanced result

Tuning parameters:

- Kmax: the order of the maximum fourier approximation
- Sigma s: spatial kernel radius of gaussian filter, controls the smoothing range
- Sigma r: controls the pixel intensity level on the filtered output

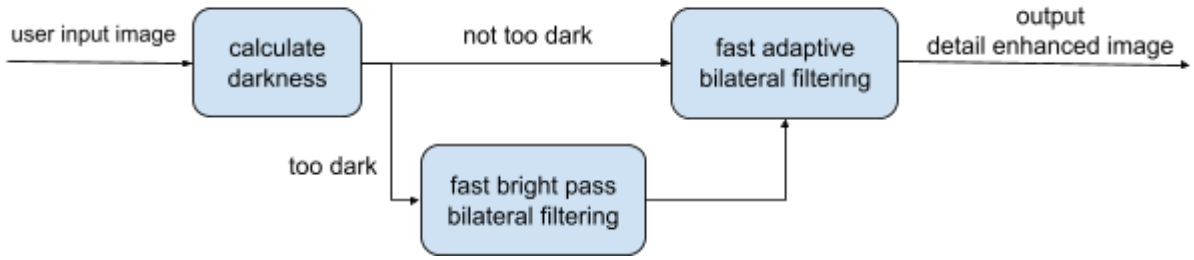
02. Fast Adaptive Bilateral Filtering of Color Images

a. Use the local covariance to determine this optimal direction
b. Approximate the local histogram by a uniform distribution along the direction of the dominant eigenvector
c. Use histogram approximation to transform the equations into integrals
d. Derive closed-form solutions for the integrals and enable the fast computation of the filtered output

Tuning parameters:

- Rho: standard deviation of gaussian spatial kernel or radius of box kernel

03. Combined



a. Check Darkness
   i. Extract the V channel
   ii. Calculate and normalize the histogram of V channel and use it to compute the cumulative distribution
   iii. Find the intensity level where the dark pixel percentage=70 is met. Check whether the level is below the low-light threshold=80 (normalized to the V channel's range 0-1) or not.
b. Fast BPBF: follow Algorithm 01
c. Fast Adaptive bilateral filtering: follow Algorithm 02

VI. Result and Discussion

We have run the algorithms on 18 images, including the ones taking outdoors and the ones taking indoors. Also, we have all resized the image to width=800 to process the combined algorithm to meet the same judging standards.

01. Evaluation Metric

a. process time: the overall enhancement time, shorter better
b. PSNR: the signal and noise compared to the original, higher better
c. PIQE: no reference metric to evaluate the visibility of details and edges, range from 0 to 100, lower better

(note: original image only calculates the PIQE metric)

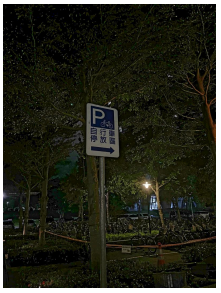
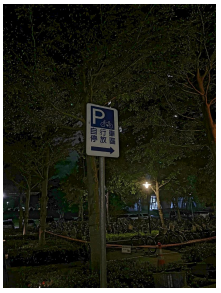
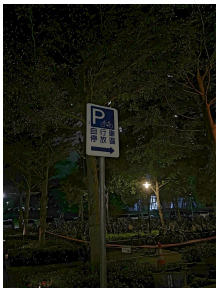
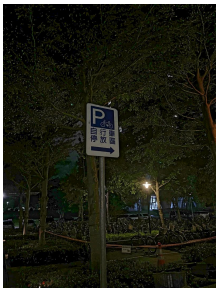
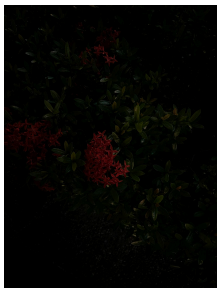
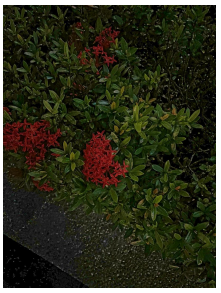
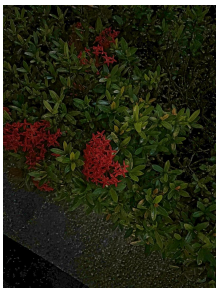
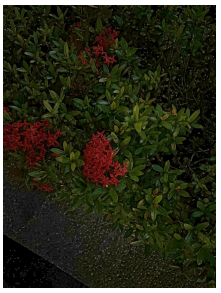
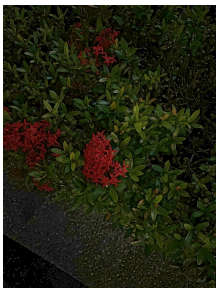
02. Different Parameters in low-light enhancement

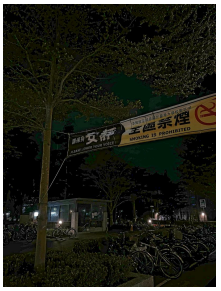
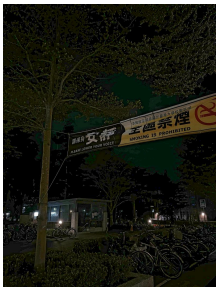
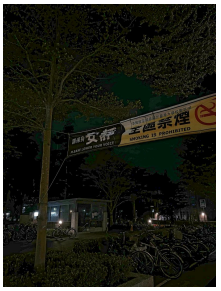
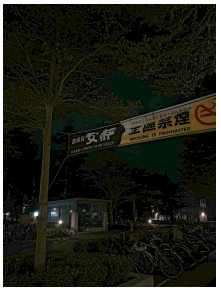
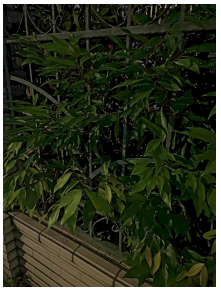
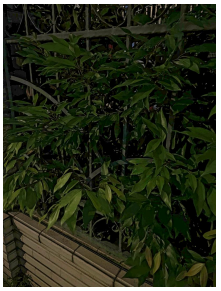
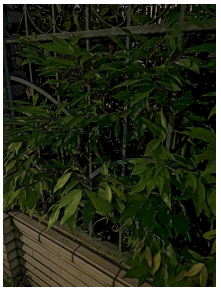
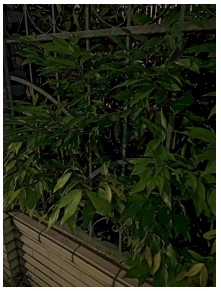
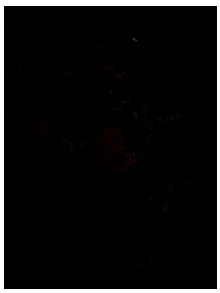
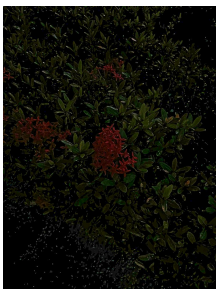
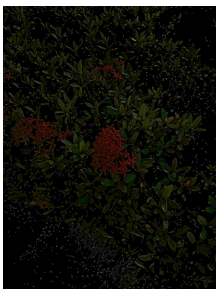
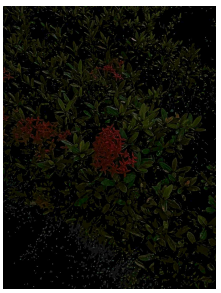
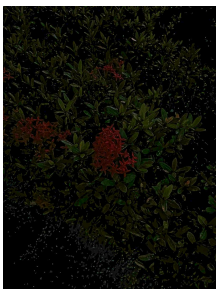
a. Kmax

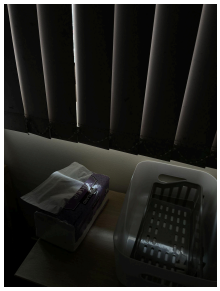
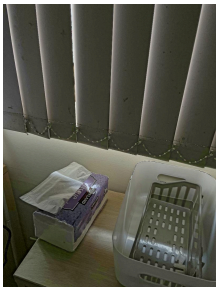
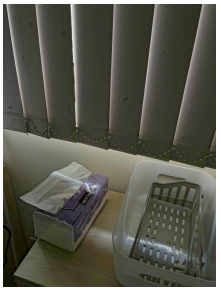
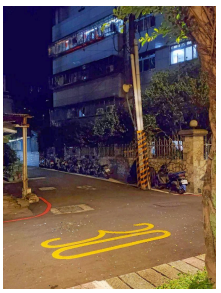
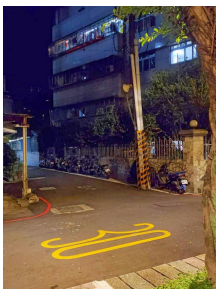
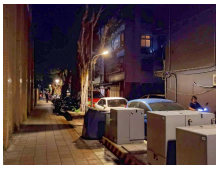
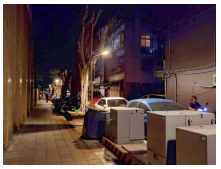
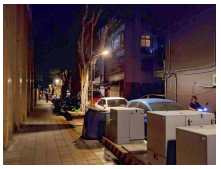
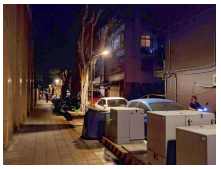
Below is the result of different Kmax parameters and their following results. We have tried with Kmax = 5, 15, 25, 35. The Kmax decides the maximum Fourier approximation order. With more order, the iterations are done more times to find the optimal parameters and expansion which results in longer processing time.

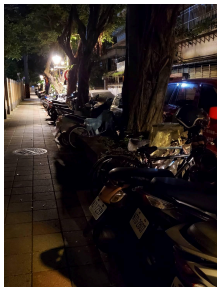
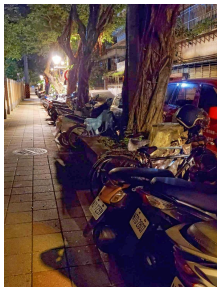
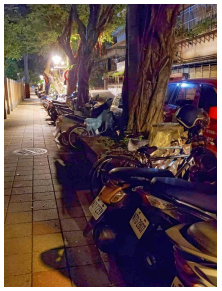
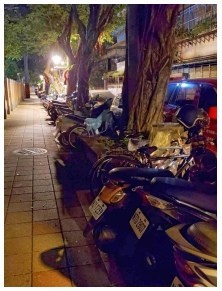
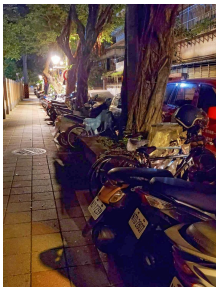
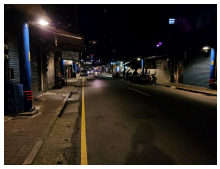
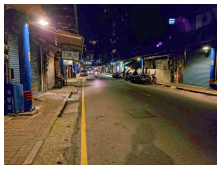
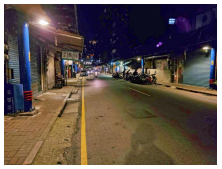
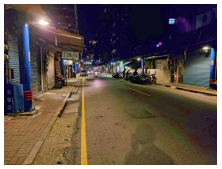
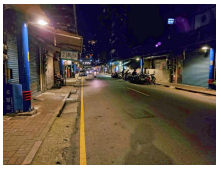
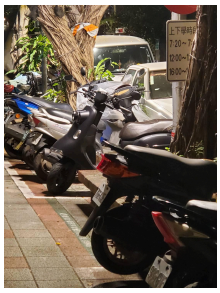
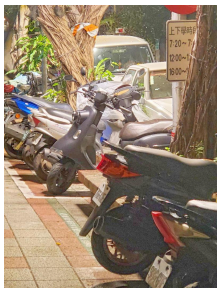
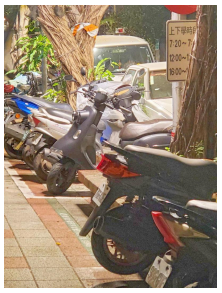
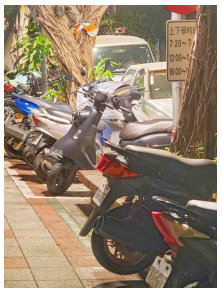
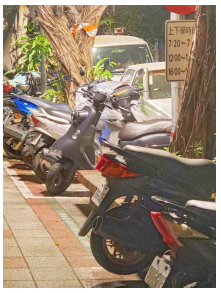
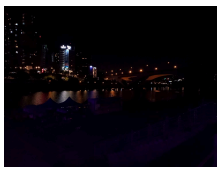
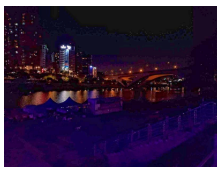
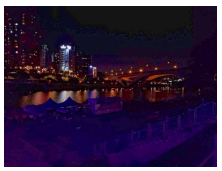
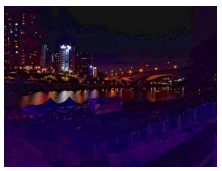
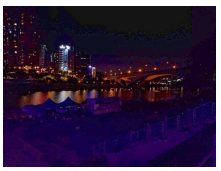
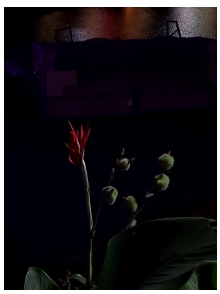
Observing the result of different Kmax values, we can see that there is only merely a difference between PSNR and PIQE. However, processing time could be extremely long when we have a larger Kmax.

We would conclude that Kmax mainly affects the processing time and has small effects on the enhancement performance. To stick with our short processing time goal, we choose **Kmax = 5** as our final param.

original	Kmax = 5	Kmax = 15	Kmax = 25	Kmax = 35
 1.jpg (24.40)	 1.29 s (27.59, 28.73)	 10.64 s (27.59, **28.11**)	 32.86 s (27.59, 28.16)	 65.43 s (27.59, 28.15)
 2.jpg (59.96)	 1.24 s (29.29, 51.00)	 10.79 s (29.31, 50.39)	 34.08 s (29.31, 50.18)	 62.37 s (29.31, **50.15**)
 3.jpg (38.72)	 1.42 s (28.56, 26.77)	 9.28 s (28.59, **25.69**)	 34.67 s (28.59, 25.73)	 65.01 s (28.59, 25.70)
 4.jpg (53.11)	 1.22 s (28.53, 49.82)	 9.45 s (28.54, 49.64)	 32.69 s (28.54, 49.61)	 67.06 s (28.54, **49.52**)

 5.jpg (64.74)	 1.22 s (29.73, 56.20)	 9.93 s (29.74, **55.73**)	 33.85 s (29.74, 56.02)	 73.60 s (29.74, 56.11)
 6.jpg (58.14)	 1.25 s (28.98, 40.69)	 9.49 s (28.99, **40.16**)	 33.56 s (28.99, 40.17)	 76.84 s (28.99, 40.32)
 7.jpg (74.32)	 1.24 s (32.24, 64.38)	 9.97 s (32.25, **64.03**)	 31.88 s (32.25, 63.83)	 65.10 s (32.25, 63.79)
 8.jpg (76.55)	 1.74 s (27.37, **70.52**)	 10.33 s (27.37, 73.92)	 32.05 s (27.37, 75.06)	 72.89 s (27.37, 75.03)

 9.jpg (51.73)	 1.26 s (27.87, 53.17)	 10.71 s (27.87, 53.21)	 32.74 s (27.86, 53.08)	 68.09 s (27.86, **53.07**)
 10.jpg (34.41)	 1.26 s (27.58, 29.15)	 10.04 s (27.51, **28.89**)	 32.68 s (27.51, 29.19)	 75.45 s (27.51, 29.17)
 11.jpg (43.62)	 1.27 s (27.75, 36.54)	 10.34 s (27.75, **36.09**)	 33.27 s (27.75, 35.90)	 67.33 s (27.75, 35.89)
 12.jpg (44.53)	 1.27 s (27.77, **37.96**)	 10.56 s (27.77, 38.55)	 32.70 s (27.77, 38.58)	 66.33 s (27.77, 38.67)
 13.jpg (62.87)	 1.13 s (27.69, **55.73**)	 8.00 s (27.69, 56.82)	 30.09 s (27.68, 56.96)	 61.45 s (27.68, 56.74)

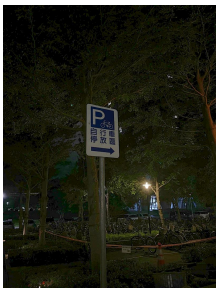
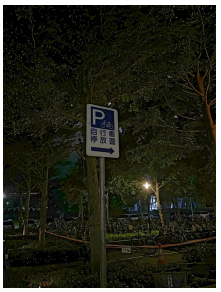
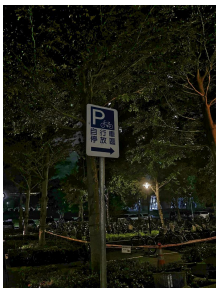
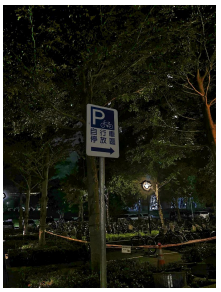
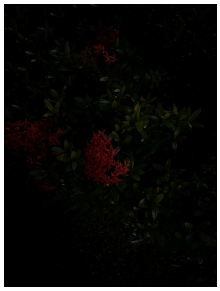
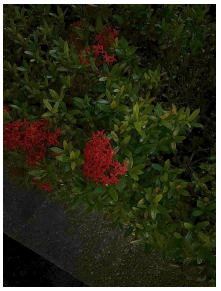
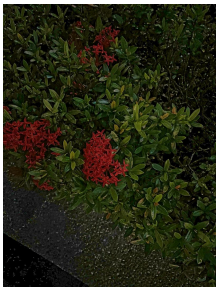
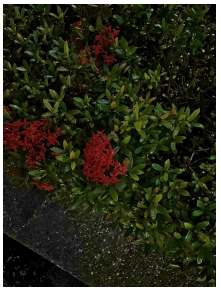
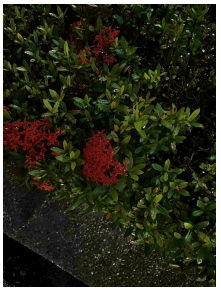
 14.jpg (38.94)	 1.27 s (27.68, 33.64)	 10.44 s (27.68, 33.28)	 33.24 s (27.68, **33.25**)	 64.39 s (27.68, **33.25**)
 15.jpg (41.77)	 0.79 s (27.63, 40.59)	 9.24 s (27.63, 40.11)	 30.98 s (27.63, 40.13)	 61.84 s (27.63, **40.01**)
 16.jpg (37.54)	 1.24 s (27.59, **26.98**)	 10.38 s (27.59, 27.82)	 31.31 s (27.59, 27.53)	 65.03 s (27.59, 27.58)
 17.jpg (62.56)	 0.78 s (28.27, 62.30)	 8.86 s (28.27, 62.21)	 31.34 s (28.27, **61.92**)	 61.18 s (28.27, **61.92**)
 18.jpg (50.42)	 1.24 s (28.12, **56.55**)	 10.17 s (28.13, 57.16)	 32.21 s (28.13, 57.10)	 62.53 s (28.13, 57.11)

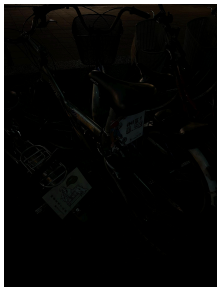
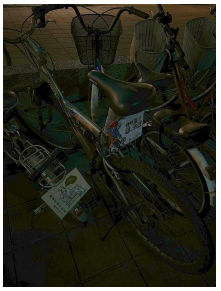
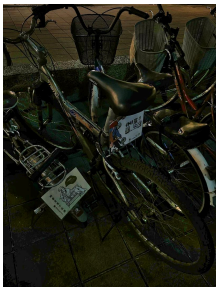
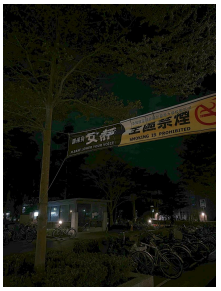
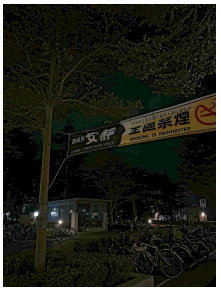
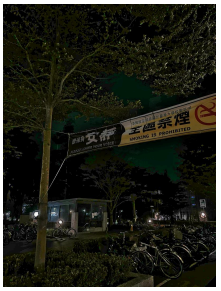
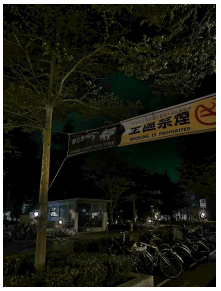
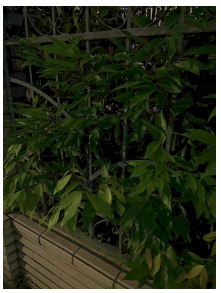
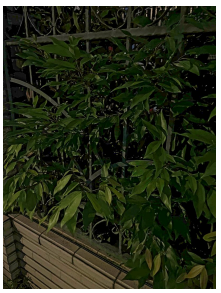
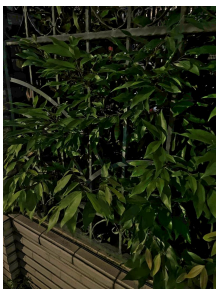
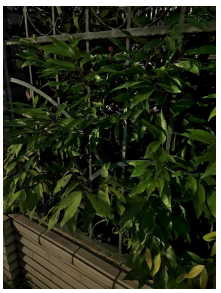
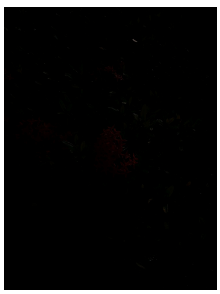
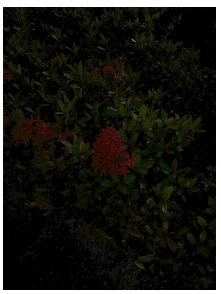
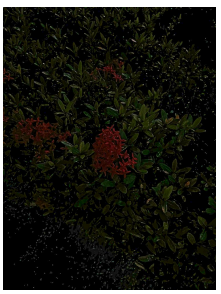
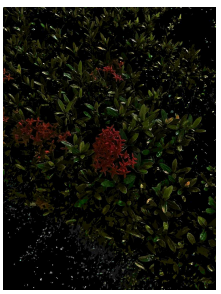
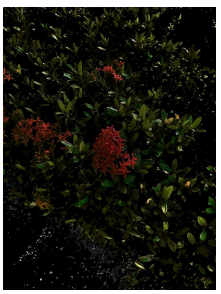
▲ different Kmax parameters result with the processing time and (PSNR, PIQE)
value in bold are the highest among the same image

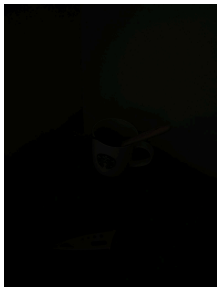
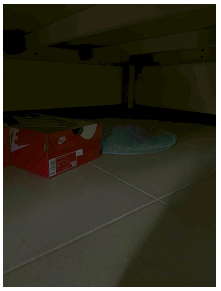
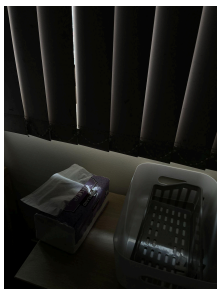
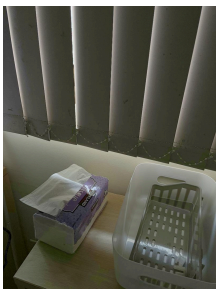
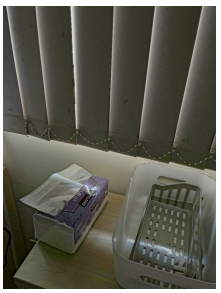
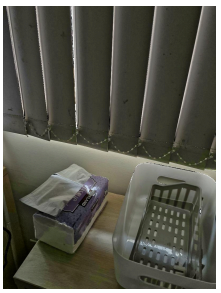
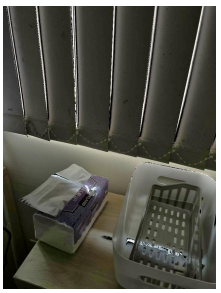
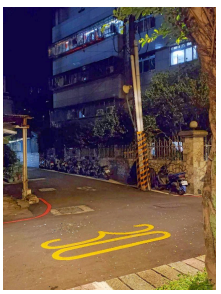
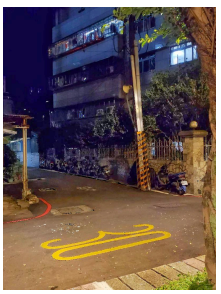
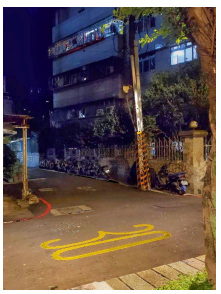
b. sigma s

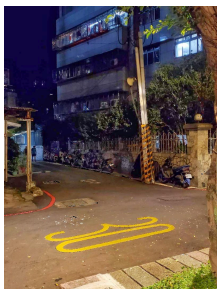
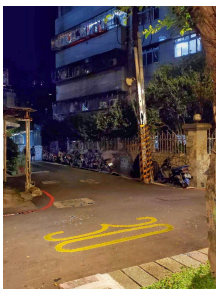
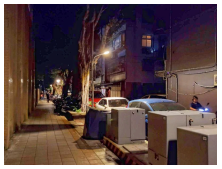
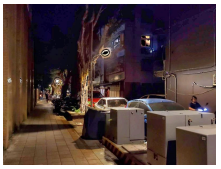
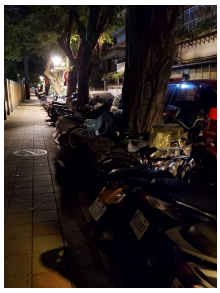
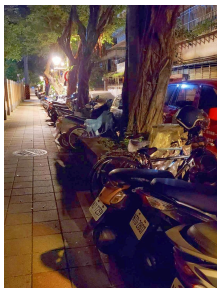
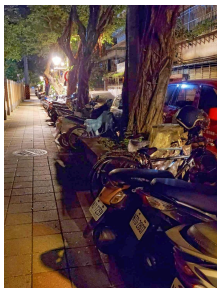
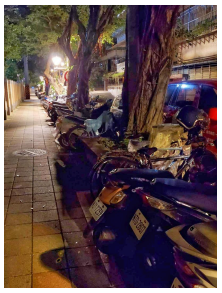
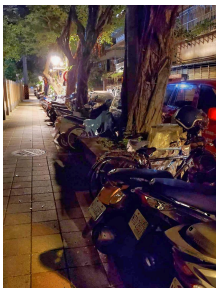
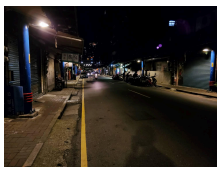
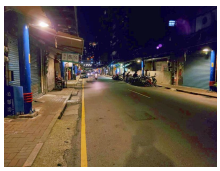
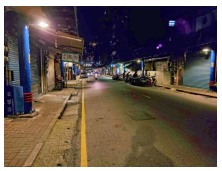
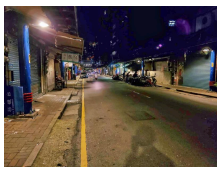
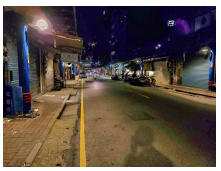
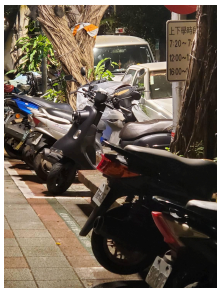
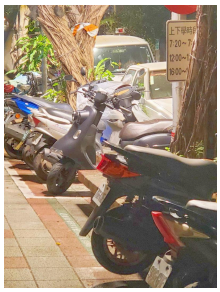
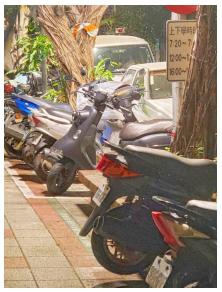
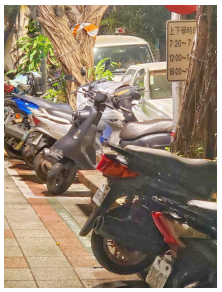
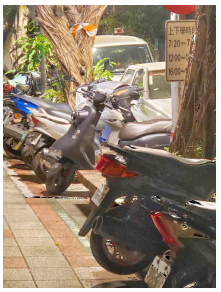
Below is the result of different sigma_s parameters. We have tried with sigma_s = 1, 5, 20, 50. The sigma_s decides the smoothing range of the objects which is the radius of the spatial kernel.

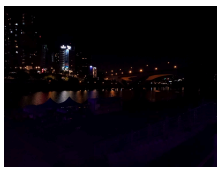
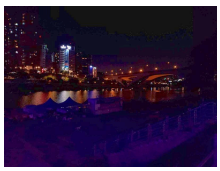
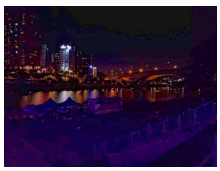
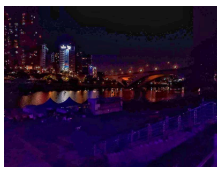
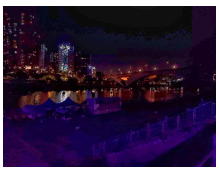
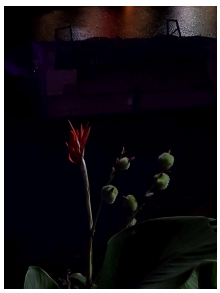
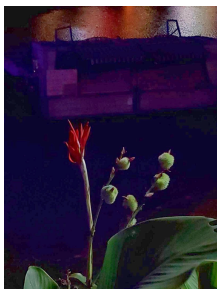
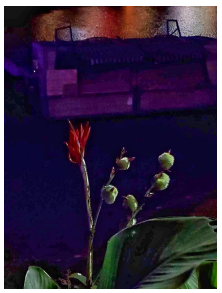
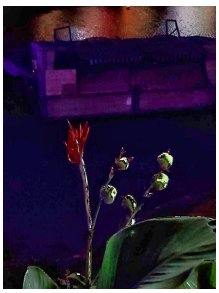
Different sigma_s results in different behaviors and values in the metric. Almost all the 18 images gain better points in PSNR and PIQE metric in either sigma_s = 50 or sigma_s = 5 cases. However, we can note that higher sigma_s also results in longer processing time which is not desirable in our goal. For some more special characteristics, we can find some black area or noise in the sigma_s = 50 case, like the characters in “5.jpg”. This could make the image have unexpected behavior in the enhanced result. In order to have a relevantly better performance and rational processing time, we chose **sigma_s = 5**.

original	sigma s = 1	sigma s = 5	sigma s = 20	sigma s = 50
				
1.jpg (24.40)	0.88 s (**27.59**, 31.83)	1.27 s (**27.59**, 28.73)	3.19 s (27.61, 27.42)	7.34 s (27.61, **27.10**)
				
2.jpg (59.96)	0.88 s (29.23, 56.09)	1.64 s (29.29, **51.00**)	3.15 s (29.34, 51.52)	7.33 s (**29.41**, 52.18)
				
3.jpg (38.72)	1.17 s (28.47, 42.11)	1.25 s (28.56, 26.77)	3.44 s (28.69, **25.55**)	7.68 s (**28.80**, 25.64)

 4.jpg (53.11)	 0.86 s (28.43, 52.27)	 1.25 s (28.53, **49.82**)	 3.17 s (28.57, 51.97)	 7.52 s (**28.63**, 52.95)
 5.jpg (64.74)	 0.85 s (29.64, 58.23)	 1.23 s (29.73, **56.20**)	 3.53 s (**29.90**, 57.36)	 7.25 s (29.88, 57.78)
 6.jpg (58.14)	 0.88 s (28.87, 43.16)	 1.25 s (28.98, **40.69**)	 3.14 s (29.12, 43.71)	 7.50 s (**29.26**, 45.51)
 7.jpg (74.32)	 0.86 s (32.22, 66.43)	 1.50 s (32.24, **64.38**)	 3.16 s (**32.31**, 65.83)	 8.09 s (32.21, 66.23)

 8.jpg (76.55)	 1.20 s (27.15, 76.81)	 1.26 s (27.37, 70.52)	 3.71 s (27.64, **70.14**)	 8.41 s (**27.71**, 72.46)
 9.jpg (51.73)	 0.86 s (27.77, 67.51)	 1.22 s (27.87, **53.17**)	 3.13 s (28.11, 59.67)	 8.25 s (**28.27**, 54.74)
 10.jpg (34.41)	 0.88 s (**27.67**, **28.14**)	 1.25 s (27.58, 29.15)	 3.15 s (27.56, 28.41)	 8.31 s (27.46, 29.68)
 11.jpg (43.62)	 0.92 s (27.73, 37.55)	 1.45 s (27.75, **36.54**)	 3.70 s (27.78, 39.23)	 8.20 s (**27.82**, 41.22)

 12.jpg (44.53)	 0.92 s (27.77, 40.64)	 1.40 s (27.77, **37.96**)	 3.18 s (27.82, 41.16)	 7.26 s (**27.85**, 42.33)
 13.jpg (62.87)	 0.59 s (**27.70**, 57.70)	 0.79 s (27.69, **55.73**)	 1.85 s (27.67, 58.56)	 4.47 s (**27.70**, 61.98)
 14.jpg (38.94)	 1.26 s (27.66, 34.83)	 1.27 s (27.68, **33.64**)	 3.74 s (27.76, 36.47)	 7.26 s (**27.80**, 38.11)
 15.jpg (41.77)	 0.58 s (27.66, 43.27)	 0.81 s (27.63, **40.59**)	 1.89 s (27.65, 42.28)	 4.48 s (**27.76**, 43.97)
 16.jpg (37.54)	 0.91 s (**27.63**, **25.80**)	 1.27 s (27.59, 26.98)	 3.19 s (27.53, 29.40)	 7.28 s (27.43, 30.81)

				
17.jpg (62.56)	0.58 s (28.07, 66.18)	1.13 s (28.27, **62.30**)	1.91 s (28.46, 66.63)	4.43 s (**28.48**, 67.58)
				
18.jpg (50.42)	0.88 s (28.06, 58.13)	1.52 s (28.12, **56.55**)	3.53 s (28.14, 57.34)	7.62 s (**28.20**, 59.15)

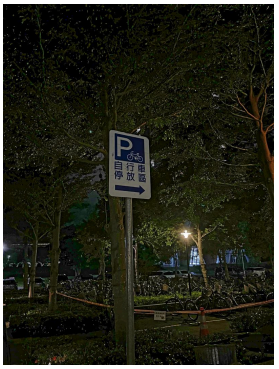
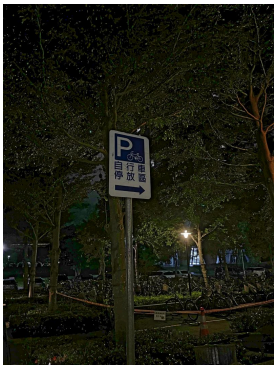
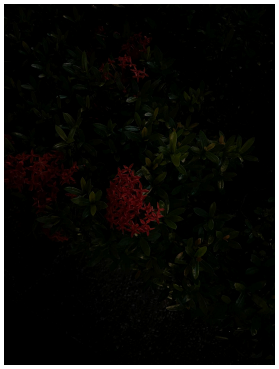
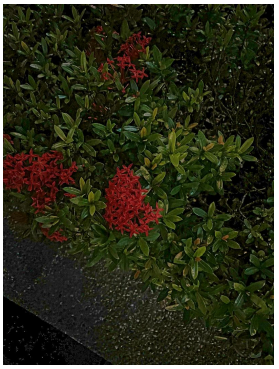
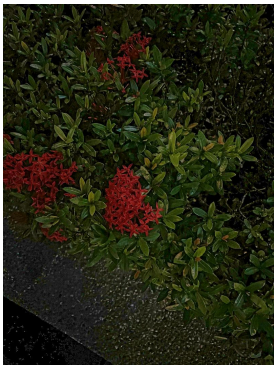
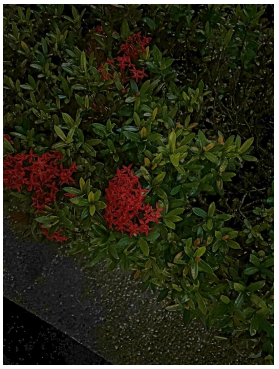
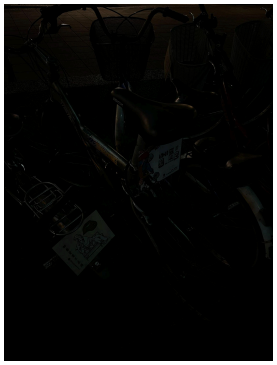
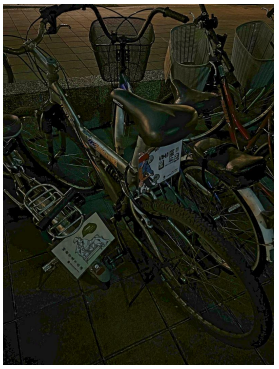
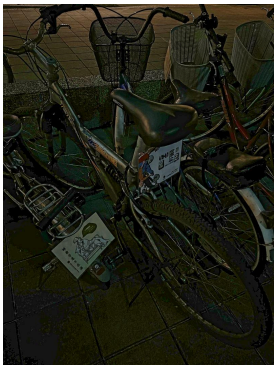
▲ different sigma_s parameters results with the processing time and (PSNR, PIQE); value in bold are the highest among the same image

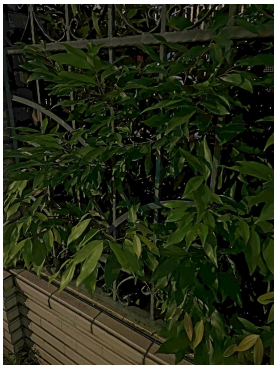
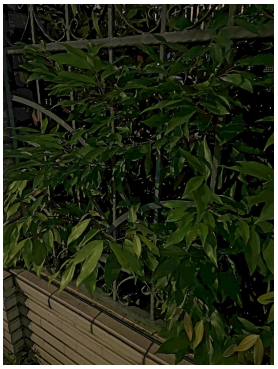
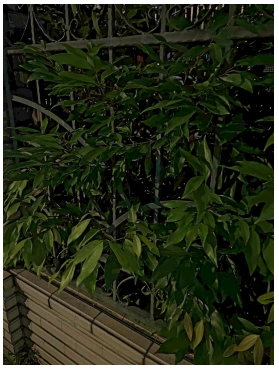
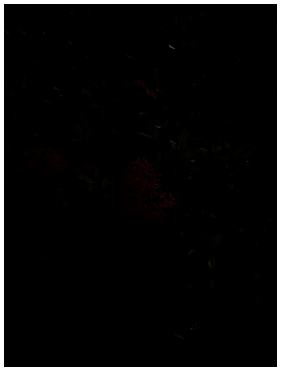
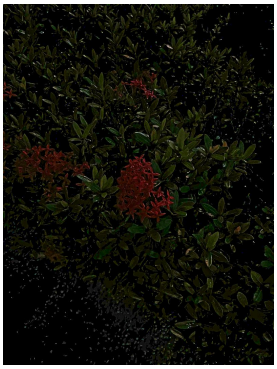
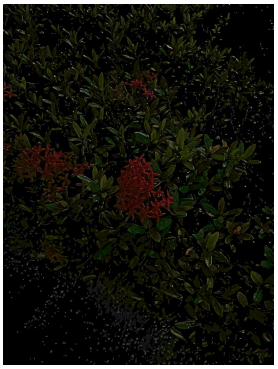
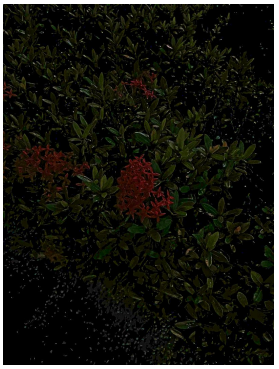
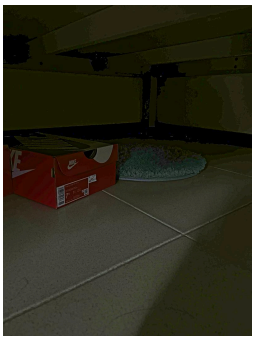
c. sigma r

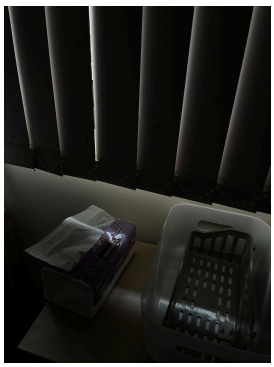
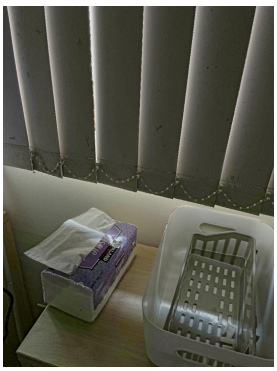
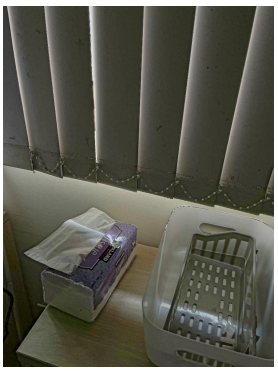
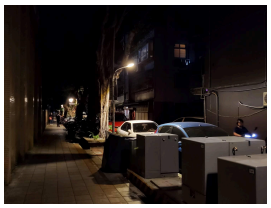
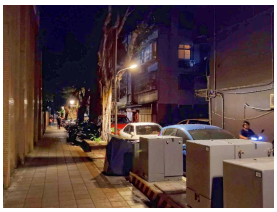
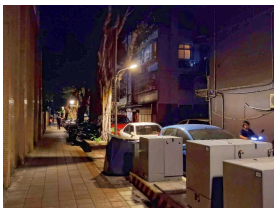
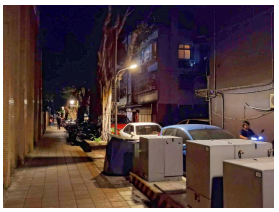
Below is the result of different sigma_r parameters. We have tried different combinations of sigma_r = 10, 30, 50. Sigma_r decides how much the pixel intensifies the filtered output from bilateral filtering.

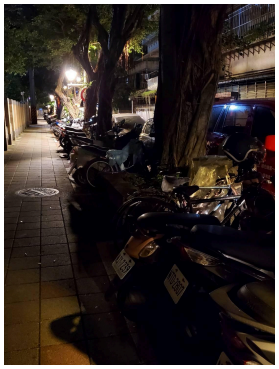
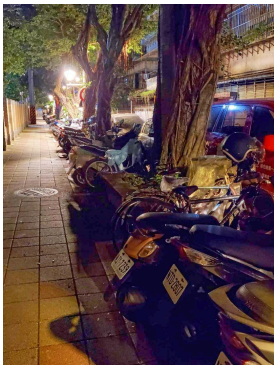
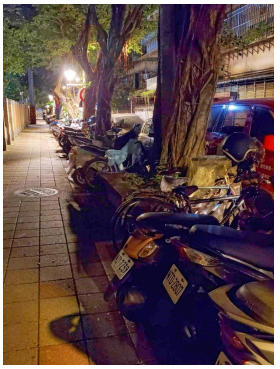
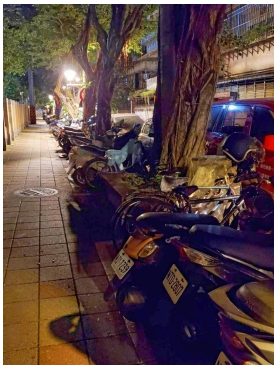
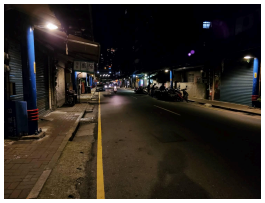
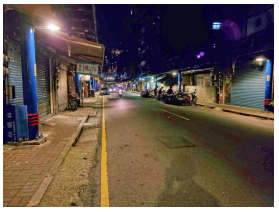
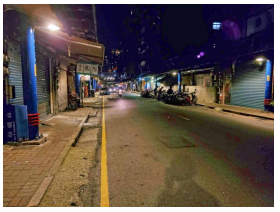
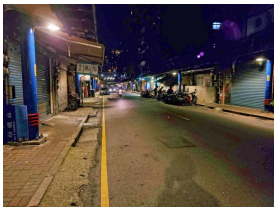
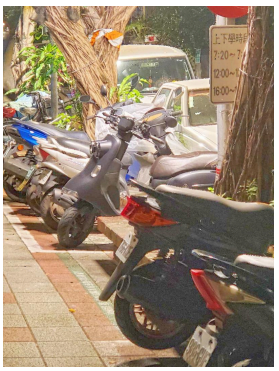
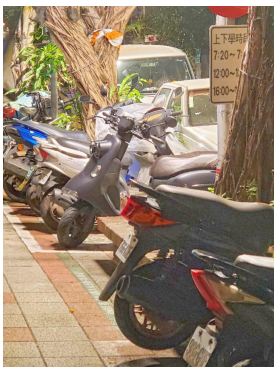
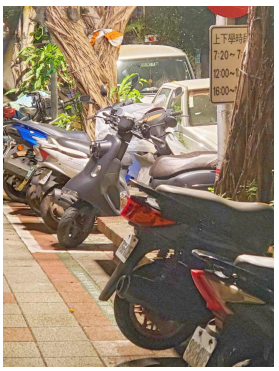
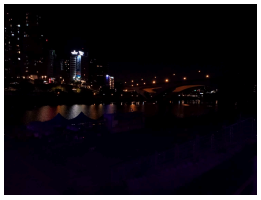
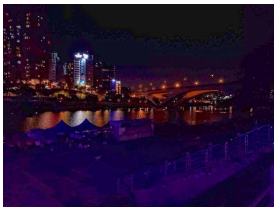
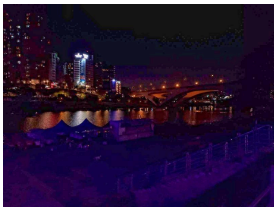
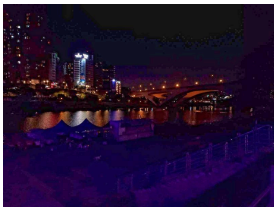
From the results, we can find that for all 18 images don't come with the same preferred param value. They all come with very close scores. Also, we can't find a big difference in the processing time and the image result with our bare eyes. To achieve an average result from the algorithm based on brightness, performance, and process time, we finally use the **sigma_r = 30** in our later approach.

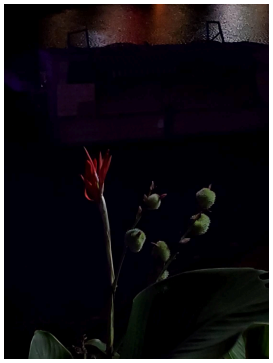
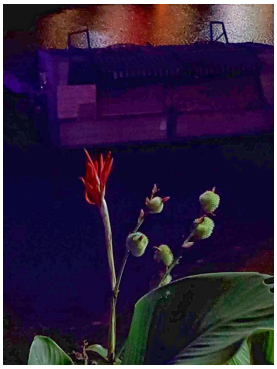
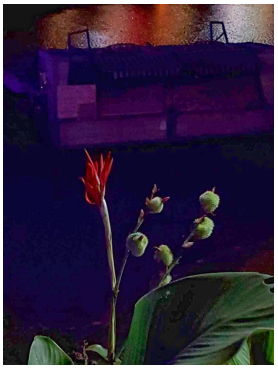
original	sigma r = 10	sigma r = 30	sigma r = 50
			
1.jpg (24.40)	1.30 s (27.58, 28.83)	1.27 s (**27.59**, 28.73)	1.64 s (**27.59**, **28.68**)

 2.jpg (59.96)	 1.26 s (29.26, 51.16)	 1.25 s (29.29, **51.00**)	 1.26 s (**29.31**, 51.15)
 3.jpg (38.72)	 1.28 s (28.53, 26.39)	 1.28 s (28.56, 26.77)	 1.25 s (**28.58**, **26.36**)
 4.jpg (53.11)	 1.87 s (28.50, 50.25)	 1.72 s (28.53, **49.82**)	 1.23 s (**28.55**, 49.90)
 5.jpg (64.74)	 1.87 s (29.69, 56.45)	 1.26 s (29.73, **56.20**)	 1.61 s (**29.75**, 56.49)

 6.jpg (58.14)	 1.87 s (28.94, **40.49**)	 1.27 s (28.98, 40.69)	 1.35 s (**28.99**, 41.16)
 7.jpg (74.32)	 1.22 s (32.24, 64.56)	 1.24 s (32.24, **64.38**)	 1.24 s (32.24, 64.40)
 8.jpg (76.55)	 1.23 s (27.36, **69.74**)	 1.47 s (**27.37**, 70.52)	 1.28 s (**27.37**, 70.55)
 9.jpg (51.73)	 1.26 s (**27.88**, 53.50)	 1.41 s (27.87, **53.17**)	 1.25 s (27.87, 53.40)

 10.jpg (34.41)	 1.26 s (27.61, **28.92**)	 1.27 s (27.58, 29.15)	 1.74 s (**27.53**, 29.41)
 11.jpg (43.62)	 1.67 s (27.72, 36.16)	 1.28 s (**27.75**, 36.54)	 1.31 s (**27.75**, **36.02**)
 12.jpg (44.53)	 1.27 s (27.74, 38.01)	 1.28 s (**27.77**, **37.96**)	 1.27 s (**27.77**, 38.40)
 13.jpg (62.87)	 0.79 s (27.69, **55.69**)	 0.89 s (27.69, 55.73)	 0.82 s (27.69, 56.93)

 14.jpg (38.94)	 1.26 s (27.66, 33.95)	 1.73 s (**27.68**, **33.64**)	 1.29 s (**27.68**, 34.23)
 15.jpg (41.77)	 0.79 s (27.63, **40.39**)	 0.82 s (27.63, 40.59)	 0.87 s (**27.64**, 41.03)
 16.jpg (37.54)	 1.28 s (**27.60**, **26.01**)	 1.32 s (27.59, 26.98)	 1.74 s (27.58, 28.24)
 17.jpg (62.56)	 1.12 s (28.25, 62.28)	 0.78 s (28.27, 62.30)	 0.78 s (**28.29**, **61.38**)

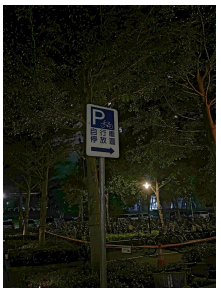
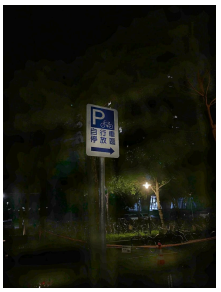
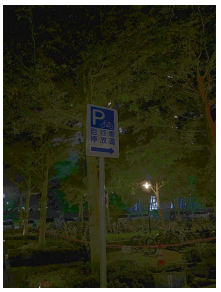
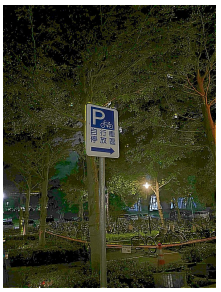
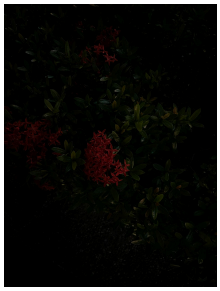
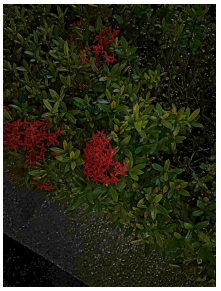
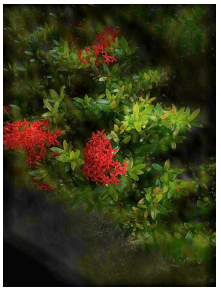
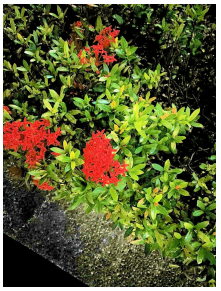
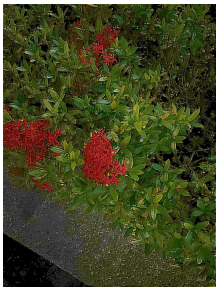
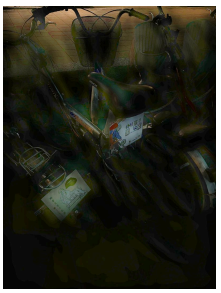
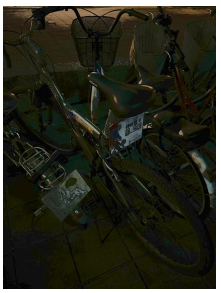
			
18.jpg (50.42)	1.27 s (28.12, **55.96**)	1.24 s (28.12, 56.55)	1.26 s (**28.13**, 56.17)

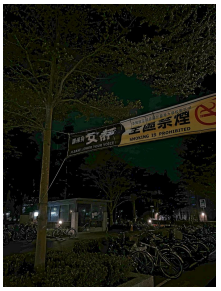
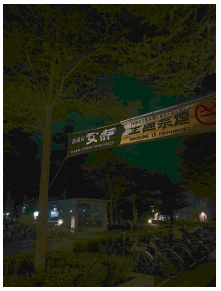
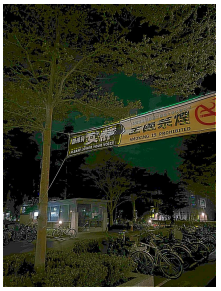
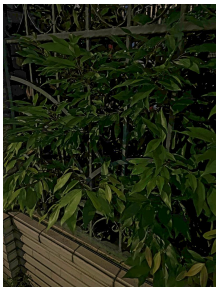
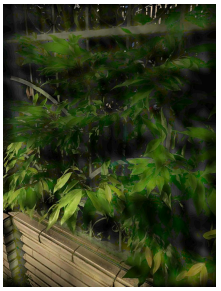
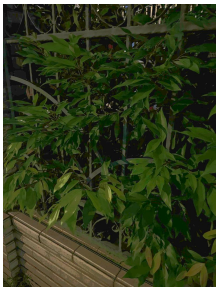
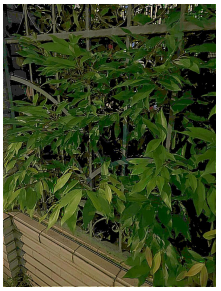
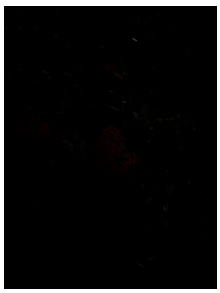
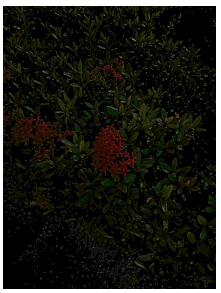
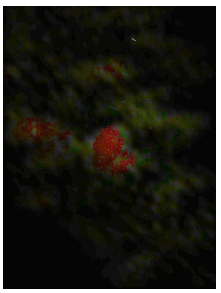
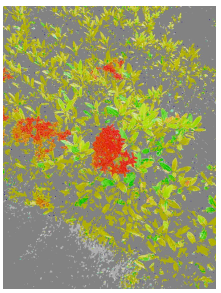
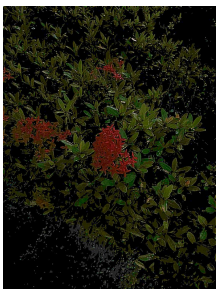
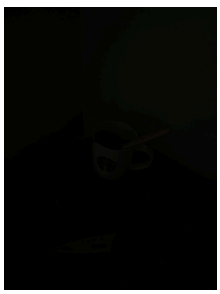
▲ different sigma_r parameters results with the processing time and (PSNR, PIQE); value in bold are the highest among the same image

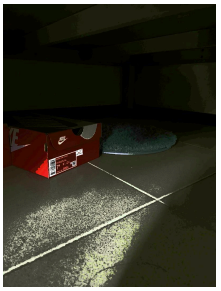
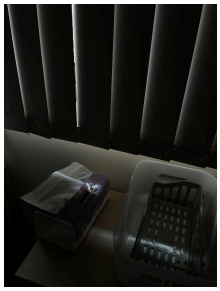
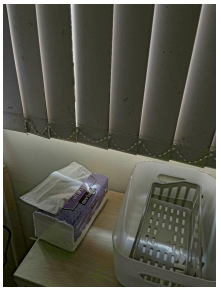
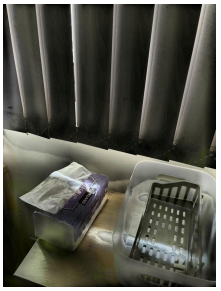
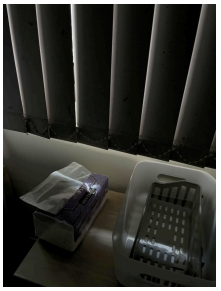
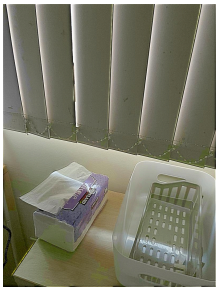
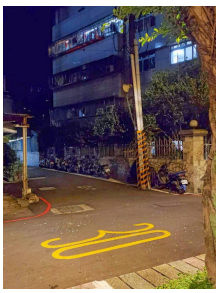
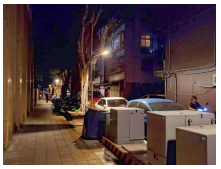
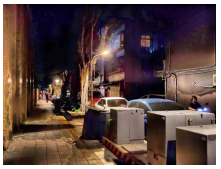
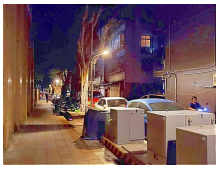
03. Compare with different low-light enhancement approaches

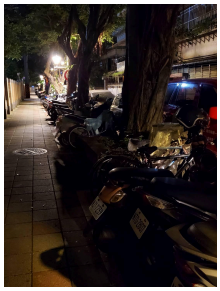
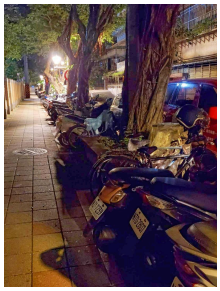
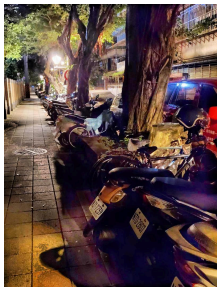
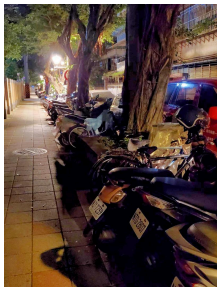
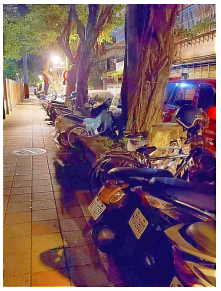
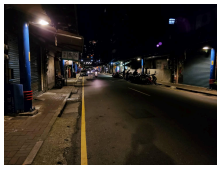
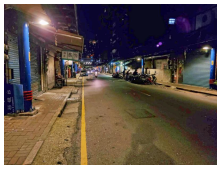
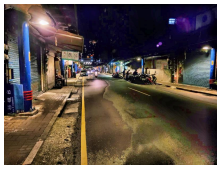
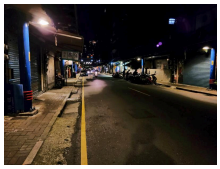
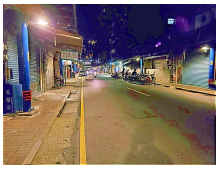
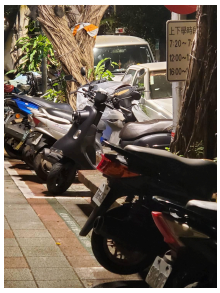
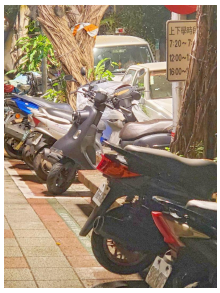
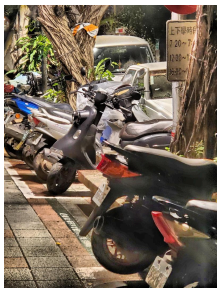
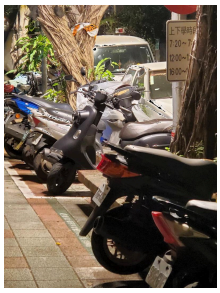
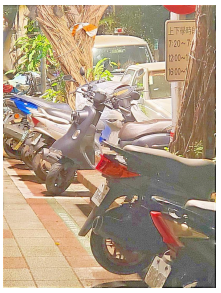
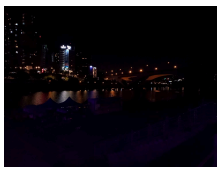
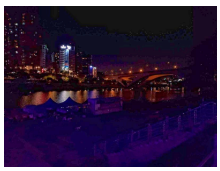
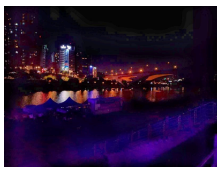
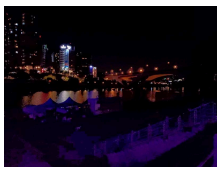
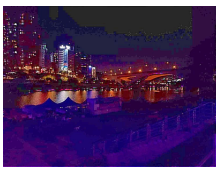
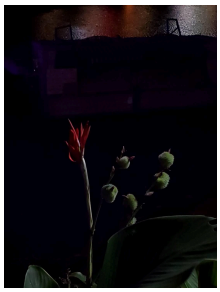
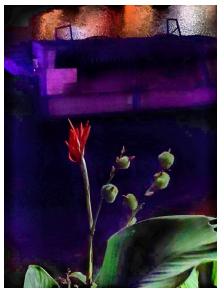
a. Fast BPBF: our implemented approach
b. Brute-force BBPBF: the BPBF iteration on every pixel based on whole image
c. BPDHE: dynamically histogram equalization
d. LIME: illumination map estimation method

Below is the result of these four approaches on all 18 images.
From generally observing all the results with bare eyes, we can see that the brute-force result images are all not focused on one point. Many background elements are shaken or blurred which is not clear enough. The brute-force also consumes much longer time than other approaches.
For BPDHE, some are enhanced excellently with bright appearance in extremely short time and gain great PSNR score. However, the “7”, “8”, “9”, “10” images that are darker or taken indoors do not get enhanced properly. They appear to lose the correct color relationship or just do not get the appropriate enhancement. Moreover, the smoothing is a bit too strong in the BPDHE results that some edges between the flowers or the object are not as clear as the fast BPBF results, like image “3” and “4”.
For LIME, we can notice that it has all enhanced the image in a recognizable way. We can all see the improvements and the objects in the picture. But there are some colors running out of balance in “5” and “8” displaying some green lights in the background which is not expected. Also, it still seems to spend much more time compared to the fast BPBF, which would not be considerable if there is going to have a bigger image size from the user.
Overall, we would like to conclude that the fast BPBF has a generally optimized result compared to the other three approaches. It could have a short but reasonable processing time, good enhanced result when observing from our eyes and applicable to all kinds of images no matter the place or any condition. This could make our proposed problem to have a desired result in the first step and would not make users feel confused or upset when receiving the enhanced image.

original	fast BPBF	brute-force	BPDHE	LIME
				
1.jpg (24.40)	1.87 s (27.59, 28.73)	92.73 s (27.70, **24.11**)	0.19 s (**28.09**, 38.61)	5.47 s (27.50, 44.70)
				
2.jpg (59.96)	1.28 s (**29.29**, **51.00**)	92.86 s (28.53, 55.38)	0.13 s (27.02, 53.09)	5.34 s (29.26, 59.55)
				
3.jpg (38.72)	1.22 s (28.56, **26.77**)	92.28 s (28.02, 27.06)	0.10 s (**29.24**, 45.34)	5.33 s (28.46, 44.37)
				
4.jpg (53.11)	1.72 s (**28.53**, **49.82**)	91.74 s (27.99, 54.45)	0.11 s (28.05, 51.07)	5.30 s (28.50, 55.74)

 5.jpg (64.74)	 1.25 s (**29.73**, **56.20**)	 91.60 s (29.28, 62.72)	 0.13 s (28.52, 56.67)	 5.20 s (29.69, 59.70)
 6.jpg (58.14)	 1.24 s (28.98, 40.69)	 92.83 s (28.01, **36.71**)	 0.10 s (**29.13**, 46.92)	 5.23 s (28.85, 43.93)
 7.jpg (74.32)	 1.28 s (**32.24**, **64.38**)	 90.89 s (28.05, 67.40)	 0.11 s (31.16, 70.29)	 5.28 s (32.21, 66.20)
 8.jpg (76.55)	 1.24 s (27.37, 70.52)	 90.48 s (**27.91**, **32.11**)	 0.05 s (27.31, 78.97)	 5.29 s (27.57, 72.14)

 9.jpg (51.73)	 1.23 s (27.87, 53.17)	 91.51 s (28.35, **42.87**)	 0.06 s (**28.38**, 66.22)	 5.35 s (27.92, 60.81)
 10.jpg (34.41)	 1.24 s (27.58, **29.15**)	 92.99 s (27.69, 42.84)	 0.13 s (**29.23**, 35.76)	 5.24 s (27.43, 41.30)
 11.jpg (43.62)	 1.82 s (27.75, 36.54)	 93.40 s (28.43, 39.71)	 0.13 s (**32.59**, 44.61)	 5.24 s (27.64, **34.90**)
 12.jpg (44.53)	 1.25 s (27.77, 37.96)	 92.96 s (28.36, 40.46)	 0.18 s (**30.88**, 45.61)	 5.27 s (27.67, **37.28**)
 13.jpg (62.87)	 0.79 s (27.69, 55.73)	 52.15 s (27.98, **53.29**)	 0.07 s (**31.53**, 62.40)	 2.37 s (27.60, 53.69)

 14.jpg (38.94)	 1.26 s (27.68, **33.64**)	 92.84 s (28.30, 36.26)	 0.13 s (**28.62,** 38.40)	 5.24 s (27.56, 35.62)
 15.jpg (41.77)	 0.79 s (27.63, **40.59**)	 52.01 s (28.19, 43.40)	 0.07 s (**32.16**, 44.51)	 2.30 s (27.41, 43.41)
 16.jpg (37.54)	 1.26 s (27.59, **26.98**)	 93.14 s (28.27, 33.27)	 0.13 s (**30.97**, 34.59)	 5.24 s (27.48, 27.83)
 17.jpg (62.56)	 0.77 s (28.27, 62.30)	 51.57 s (28.78, **55.48**)	 0.07 s (**33.93**, 66.46)	 2.28 s (28.44, 61.47)
 18.jpg (50.42)	 1.22 s (28.12, 56.55)	 91.61 s (28.54, **51.47**)	 0.10 s (**31.05**, 62.77)	 5.15 s (27.55, 57.59)

▲ different approach results with the processing time and (PSNR, PIQE);
value in bold are the highest among the same image

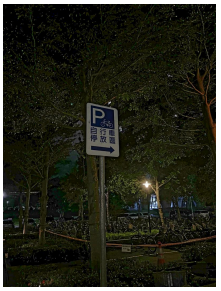
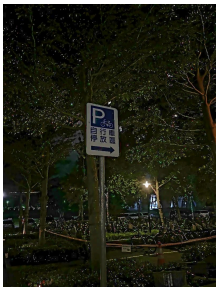
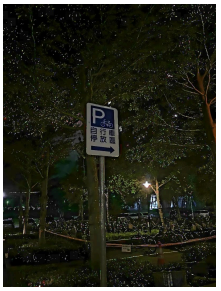
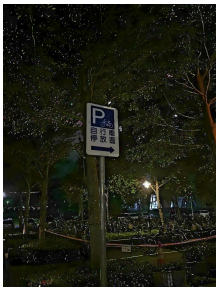
04. Different Parameters in adaptive bilateral filtering

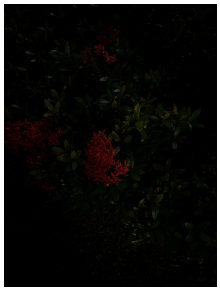
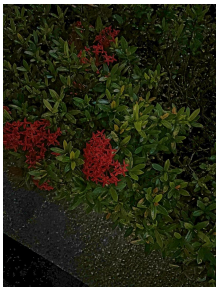
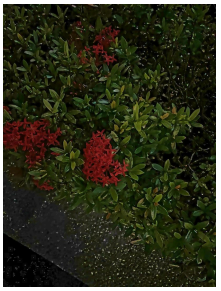
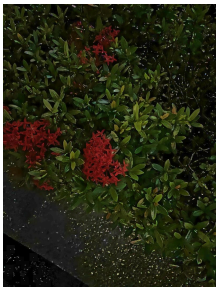
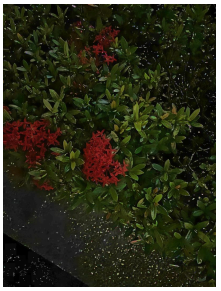
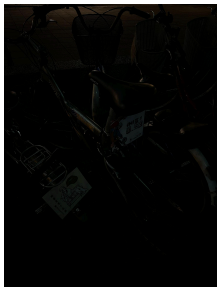
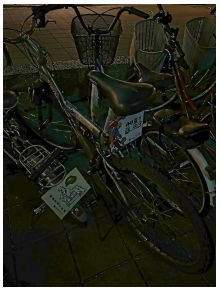
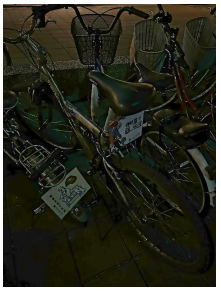
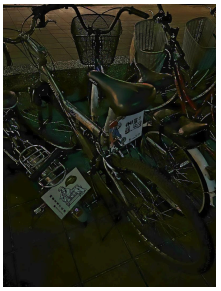
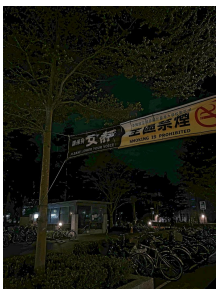
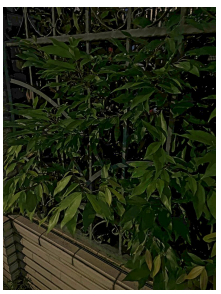
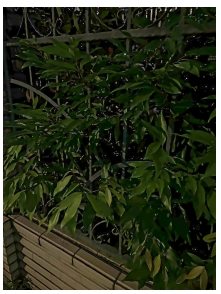
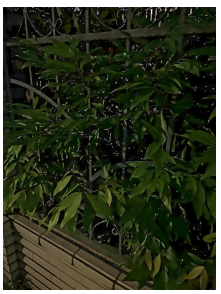
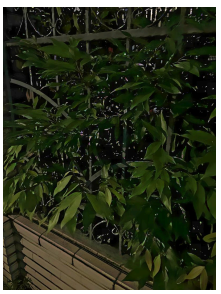
a. rho for sharpening:

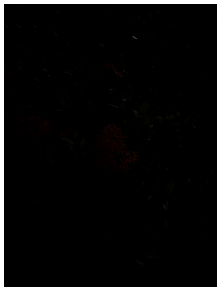
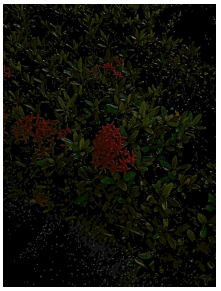
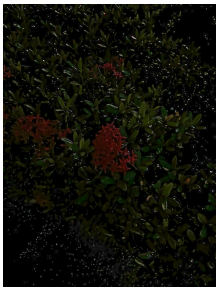
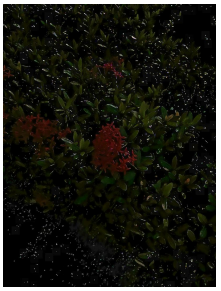
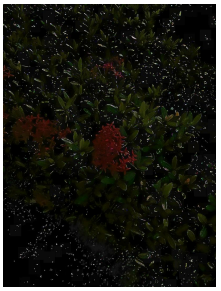
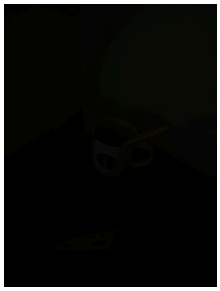
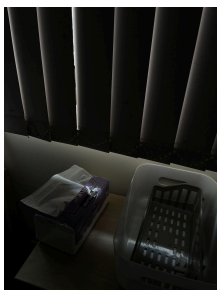
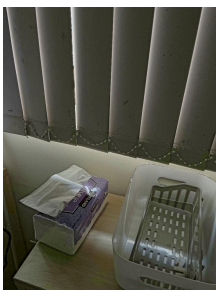
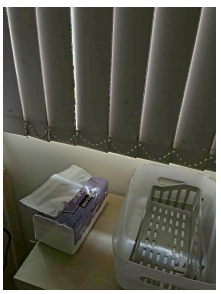
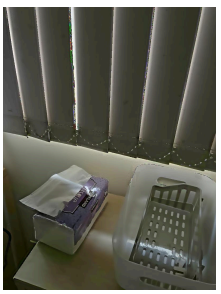
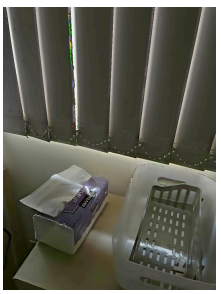
We tried several values of rho, and show the result in the below table. The outcome shows that rho = 3 performs the best. The next table further shows the differences on 18 images. In most cases, the output images seem to be similar. However, when rho becomes bigger, it is more likely to output images that contain artifacts (some noise) even though the PIQE score is better. Therefore, we select rho = 3.

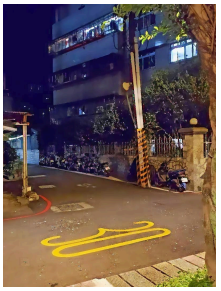
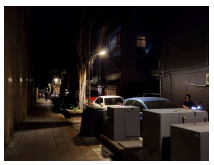
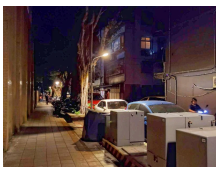
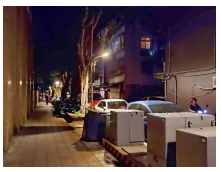
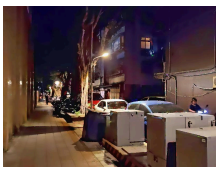
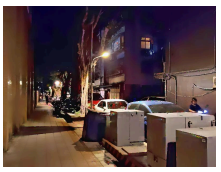
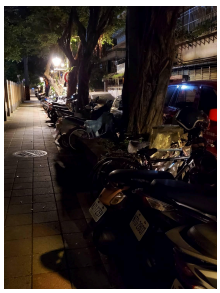
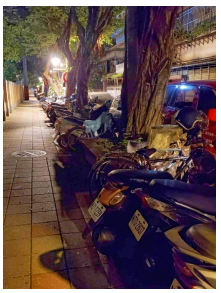
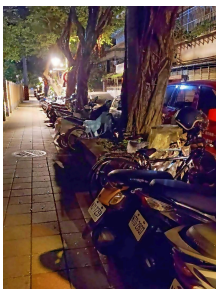
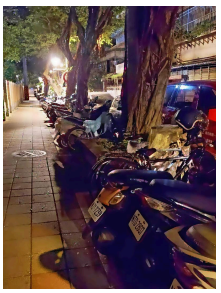
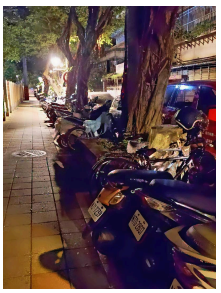
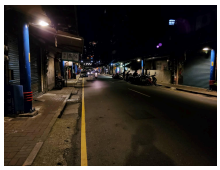
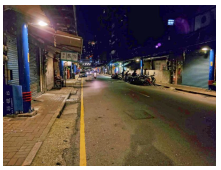
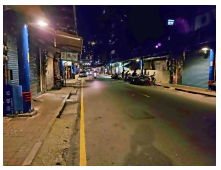
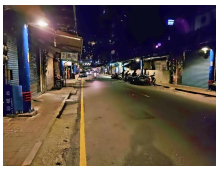
rho	PSNR (mean)	PSNR (std)	PIQE (mean)	PIQE (std)	Time (s) (mean)	Time (s) (std)
3	**34.69**	2.57	53.13	8.70	**17.82**	3.61
5	33.79	2.43	46.49	8.29	24.56	4.52
7	33.29	2.39	**44.36**	8.83	35.46	6.42

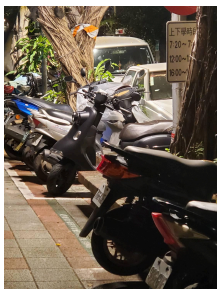
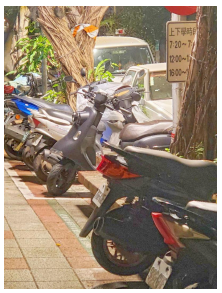
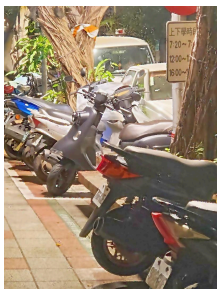
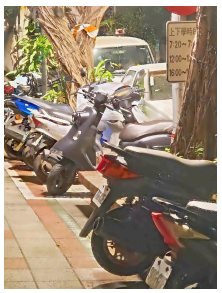
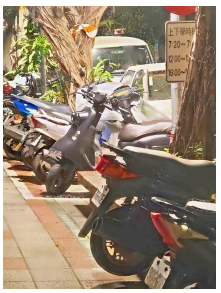
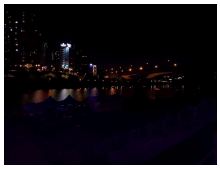
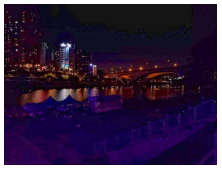
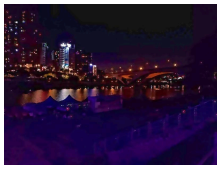
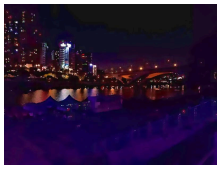
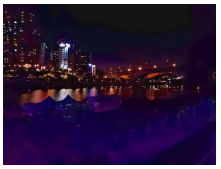
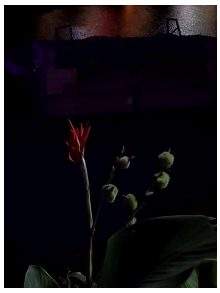
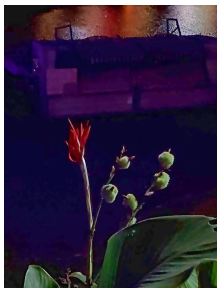
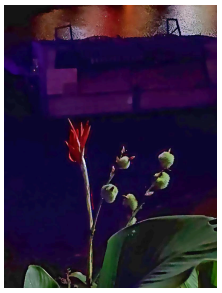
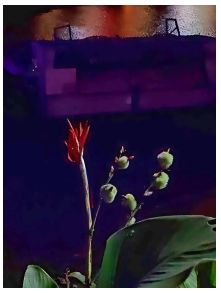
▲ Comparison between three values of rho.

original	low light enhanced	rho = 3	rho = 5	rho = 7
				
1.jpg (**24.40**)	1.29 s (27.59, 28.73)	17.78 s **(35.47,** 61.52**)**	25.56 s (34.71, 53.34)	37.10 s (34.39, 48.57)
				
2.jpg (59.96)	1.24 s (29.29, 51.00)	18.91 s (**32.12**, 46.76)	25.87 s (31.49, 42.27)	37.83 s (31.12, **40.86**)

 3.jpg (38.72)	 1.42 s (28.56, 26.77)	 18.90 s (**32.56**, 37.69)	 26.11 s (31.94, 29.50)	 40.08 s (31.61, **27.49**)
 4.jpg (53.11)	 1.22 s (28.53, 49.82)	 20.36 s (**32.95**, 46.31)	 27.39 s (32.32, 40.33)	 37.99 s (31.93, **39.74**)
 5.jpg (64.74)	 1.22 s (29.73, 56.20)	 18.65 s (**32.91**, 53.56)	 26.19 s (32.02, 48.59)	 37.39 s (31.47, **47.93**)
 6.jpg (58.14)	 1.25 s (28.98, **40.69**)	 18.60 s (**33.57**, 55.46)	 26.33 s (32.60, 46.98)	 37.75 s (31.99, 42.92)

 7.jpg (74.32)	 1.24 s (32.24, 64.38)	 18.63 s (**33.63**, 64.87)	 26.13 s (32.05, **59.20**)	 37.21 s (31.02, 60.73)
 8.jpg (76.55)	 1.74 s (27.37, 70.52)	 18.84 s (**42.33**, 70.69)	 26.09 s (41.20, **53.94**)	 43.19 s (40.59, 59.02)
 9.jpg (51.73)	 1.26 s (27.87, 53.17)	 19.76 s (**38.76**, 56.00)	 28.09 s (37.66, 53.54)	 37.87 s (37.01, **50.51**)
 10.jpg (34.41)	 1.26 s (27.58, 29.15)	 18.45 s (**36.09**, 61.79)	 25.76 s (34.90, 54.65)	 37.11 s (34.29, **49.79**)

 11.jpg (43.62)	 1.27 s (27.75, **36.54**)	 18.56 s (**33.66**, 47.35)	 25.88 s (33.08, 40.08)	 37.27 s (32.78, 37.31)
 12.jpg (44.53)	 1.27 s (27.77, **37.96**)	 18.79 s (**33.48**, 45.77)	 26.23 (32.92, 40.74)	 37.57 s (32.65, 38.30)
 13.jpg (62.87)	 1.13 s (27.69, 55.73)	 10.38 s (**34.43**, 63.90)	 14.40 s (33.37, 59.92)	 21.00 s (32.80, **54.99**)
 14.jpg (38.94)	 1.27 s (27.68, **33.64**)	 19.02 s (**33.09**, 41.91)	 26.03 s (32.46, 35.80)	 37.84 s (32.06, 34.04)
 15.jpg (41.77)	 0.79 s (27.63, **40.59**)	 10.42 s (**33.62**, 49.09)	 14.70 s (33.00, 43.65)	 21.32 s (32.62, 42.41)

 16.jpg (37.54)	 1.24 s (27.59, **26.98**)	 19.02 s (**32.15**, 44.16)	 26.57 s (31.40, 36.54)	 38.71 s (30.98, 33.49)
 17.jpg (62.56)	 0.78 s (28.27, 62.30)	 10.59 s (**36.54**, 54.72)	 14.96 s (35.28, 51.97)	 22.03 s (34.70, **51.27**)
 18.jpg (50.42)	 1.24 s (28.12, 56.55)	 25.04 s (**37.09**, 54.74)	 29.73 s (35.81, 45.72)	 39.05 s (35.15, **39.18**)

▲ different rho results after low light enhancement and after sharpening;
with the processing time and (PSNR, PIQE);
value in bold are the highest among the same image

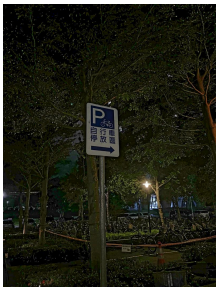
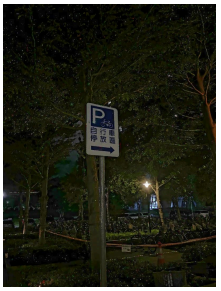
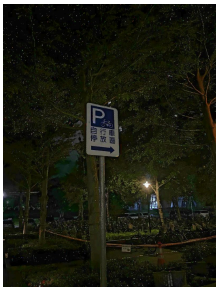
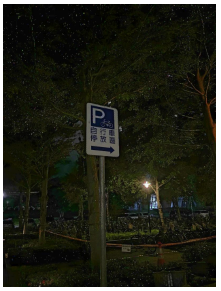
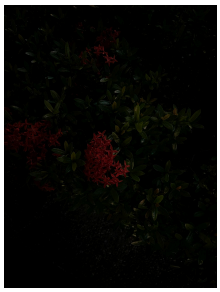
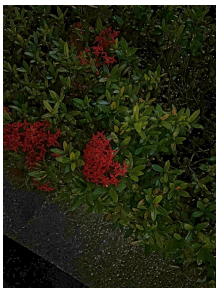
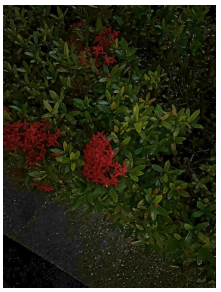
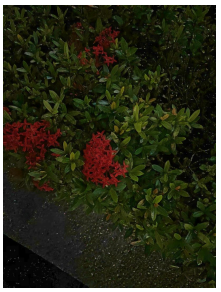
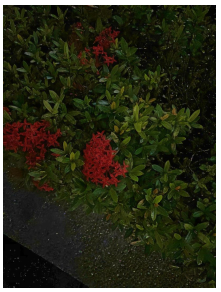
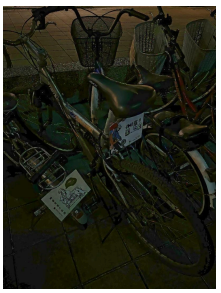
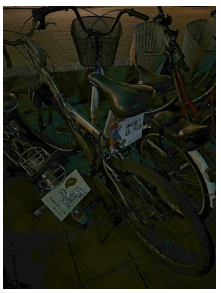
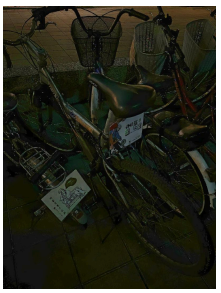
b. rho for deblocking

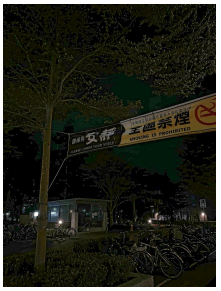
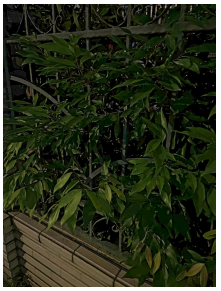
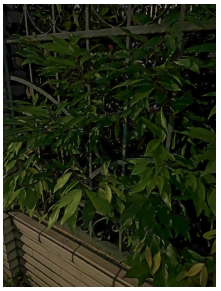
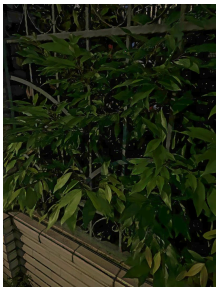
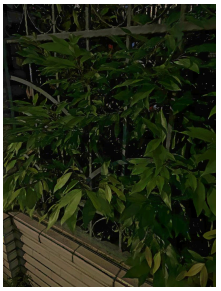
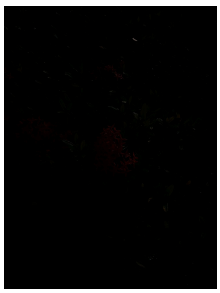
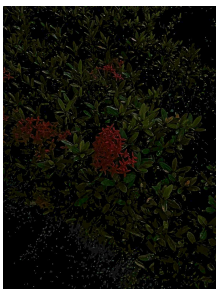
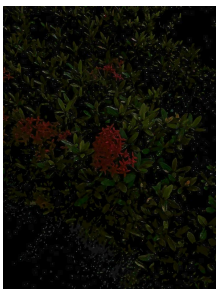
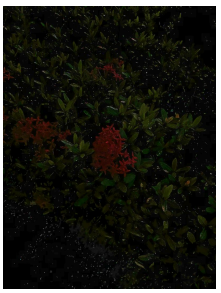
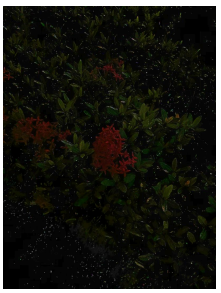
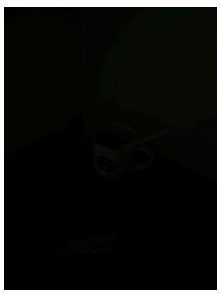
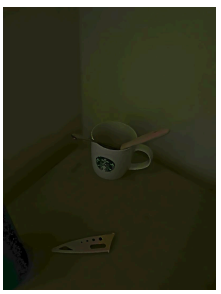
We tried several values of rho, and show the result in the below table. The outcome shows that rho = 3 performs the best. The next table further shows the differences on 18 images. The result is similar to that of sharpening. Although the PIQE is higher with bigger rho, the images may contain artifacts. Moreover, we consider the output of rho = 3 is better using humans' eyes. Besides, the time cost is relatively small. Therefore, we select rho = 3.

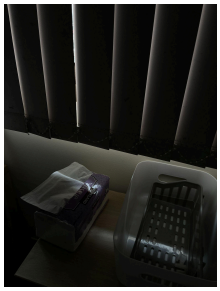
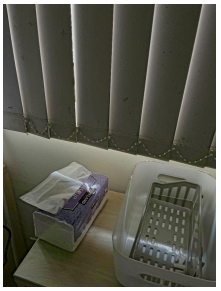
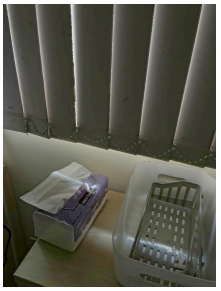
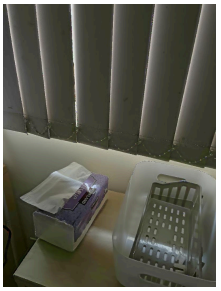
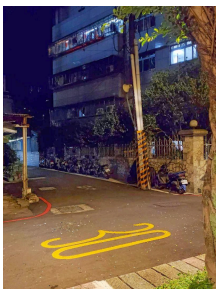
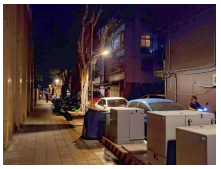
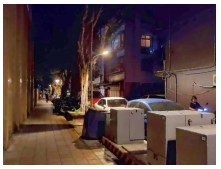
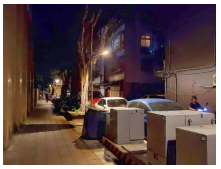
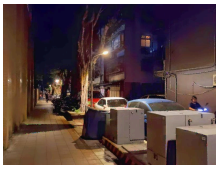
rho	PSNR (mean)	PSNR (std)	PIQE (mean)	PIQE (std)	Time (s) (mean)	Time (s) (std)
3	**35.67**	2.56	41.51	9.24	**20.42**	3.82
5	35.23	2.44	37.73	9.65	27.97	5.59

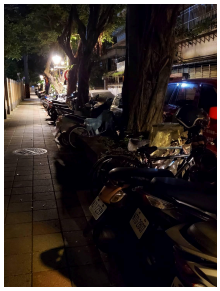
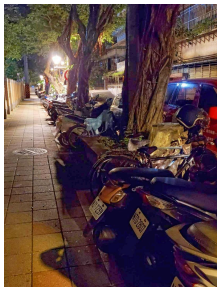
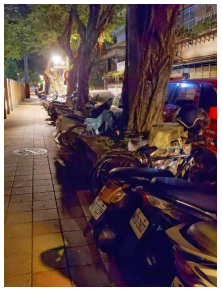
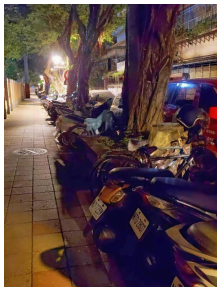
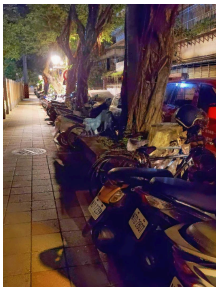
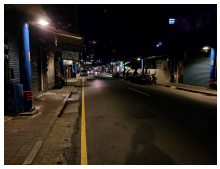
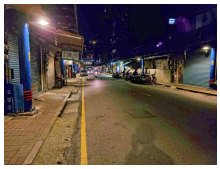
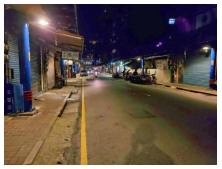
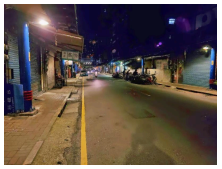
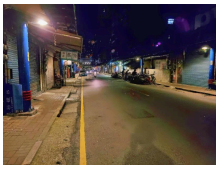
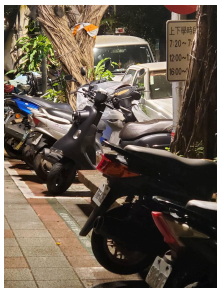
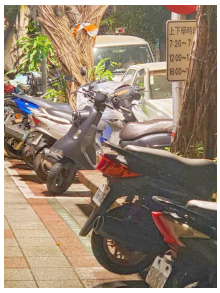
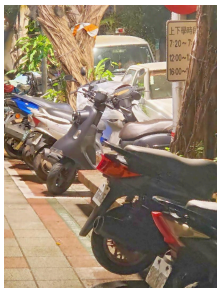
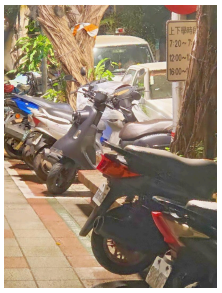
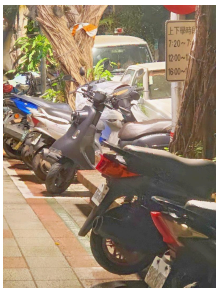
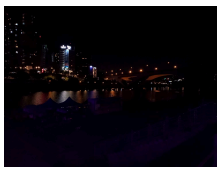
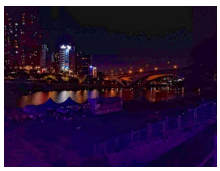
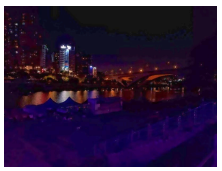
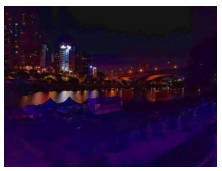
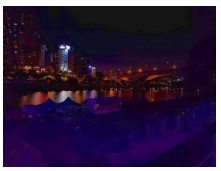
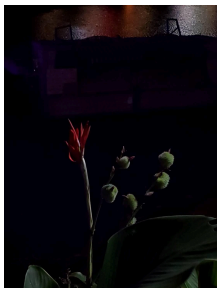
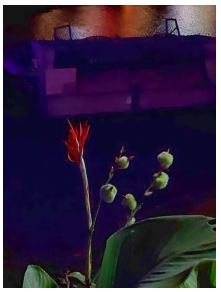
7	35.08	2.43	**36.10**	9.96	39.63	7.62

▲ Comparison between three values of rho.

original	low light enhanced	rho = 3	rho = 5	rho = 7
				
1.jpg (**24.40**)	1.29 s (27.59, 28.73)	20.29 s (**36.01**, 48.60)	29.01 s (35.60, 41.06)	41.40 s (35.60, 34.66)
				
2.jpg (59.96)	1.24 s (29.29, 51.00)	23.99 s (**32.59**, 34.53)	31.43 s (32.26, 32.97)	43.95 s (32.10, **31.97**)
				
3.jpg (38.72)	1.42 s (28.56, 26.77)	23.57 s (**33.27**, 24.02)	38.80 s (32.83, 19.55)	41.50 s (32.63, **18.62**)
				
4.jpg (53.11)	1.22 s (28.53, 49.82)	22.74 s (**33.51**, 33.72)	31.87 s (33.13, 31.78)	45.58 s (32.93, **30.46**)

 5.jpg (64.74)	 1.22 s (29.73, 56.20)	 22.32 s (**33.28**, 43.24)	 29.70 s (32.61, 40.48)	 43.29, ((32.22, **38.76**)
 6.jpg (58.14)	 1.25 s (28.98, 40.69)	 21.61 s (**34.05**, 41.22)	 29.19 s (33.24, 36.80)	 49.30 s (32.78, **34.65**)
 7.jpg (74.32)	 1.24 s (32.24, 64.38)	 21.26 s (**33.55**, 58.19)	 28.73 s (32.19, 55.74)	 41.93 s (31.47, **54.67**)
 8.jpg (76.55)	 1.74 s (27.37, 70.52)	 23.08 s (**42.87**, **55.95**)	 30.10 s (41.93, 57.09)	 42.47 s (41.45, 56.42)

 9.jpg (51.73)	 1.26 s (27.87, 53.17)	 21.65 s (**39.36**, 51.07)	 28.99 s (38.48, 44.93)	 40.94 s (38.01, **44.60**)
 10.jpg (34.41)	 1.26 s (27.58, **29.15**)	 21.35 s (**37.36**, 48.31)	 29.24 s (36.86, 43.81)	 42.59 s (36.64, 40.31)
 11.jpg (43.62)	 1.27 s (27.75, 36.54)	 21.95 s (35.15, 34.71)	 30.26 s (35.28, 29.99)	 43.02 s (**35.53**, **28.93**)
 12.jpg (44.53)	 1.27 s (27.77, 37.96)	 20.99 s (34.96, 36.37)	 28.63 s (35.09, 32.71)	 42.60 s (**35.35**, **32.46**)
 13.jpg (62.87)	 1.13 s (27.69, 55.73)	 12.37 s (**36.04**, 53.74)	 16.58 s (35.63, 48.49)	 23.62 s (35.59, **47.10**)

 14.jpg (38.94)	 1.27 s (27.68, 33.64)	 22.42 s (34.71, 31.61)	 30.93 s (34.85, 28.49)	 42.82 s (**34.95**, **27.28**)
 15.jpg (41.77)	 0.79 s (27.63, 40.59)	 12.04 s (35.18, 36.08)	 16.21 s (35.40, 32.23)	 22.69 s (**35.68**, **31.72**)
 16.jpg (37.54)	 1.24 s (27.59, 26.98)	 21.84 s (33.95, 31.86)	 28.54 s (34.18, 25.43)	 40.39 s (**34.49, 23.12**)
 17.jpg (62.56)	 0.78 s (28.27, 62.30)	 11.88 s (**37.84,** 42.70)	 16.94 s (37.03, 41.08)	 23.08 s (36.73, **39.69**)
 18.jpg (50.42)	 1.24 s (28.12, 56.55)	 22.35 s (**38.38,** 41.21)	 28.31 s (37.59, 36.49)	 42.11 s (37.25, **34.40**)

▲ different rho results after low light enhancement and after deblocking;
with the processing time and (PSNR, PIQE);
value in bold are the highest among the same image

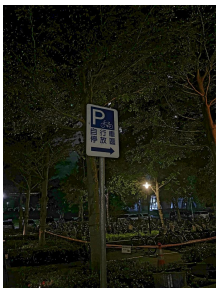
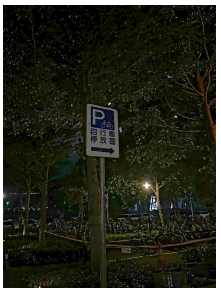
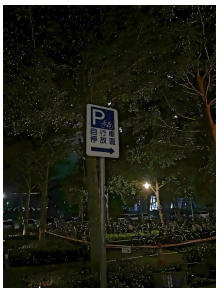
05. Compare with fast and brute force approach (take sharpening for example)

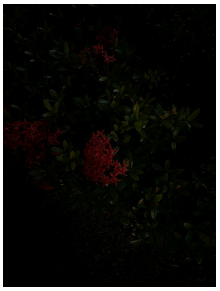
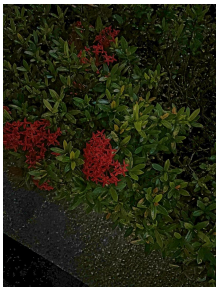
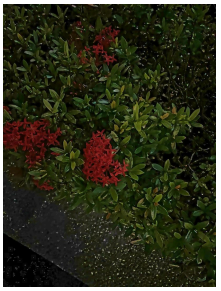
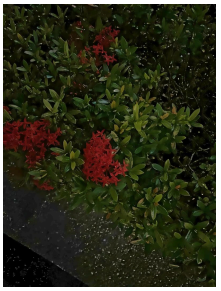
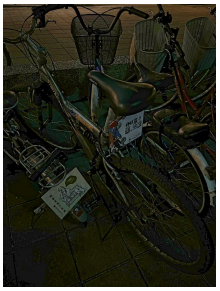
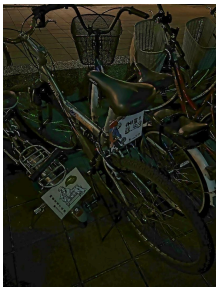
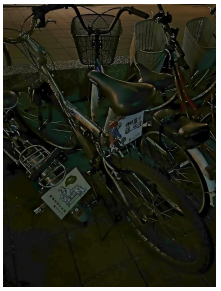
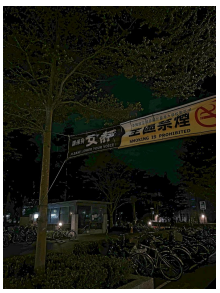
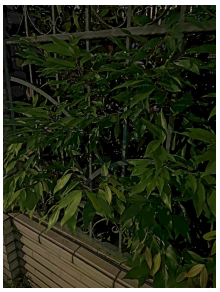
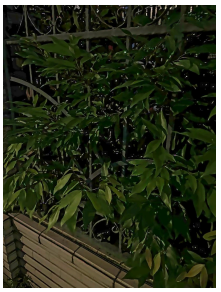
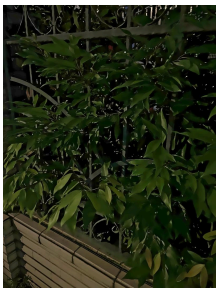
a. Fast adaptive bilateral filtering
b. Brute-force adaptive bilateral filtering

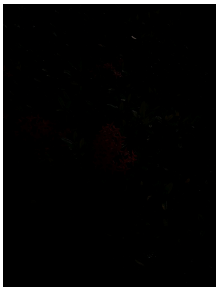
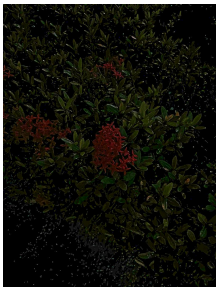
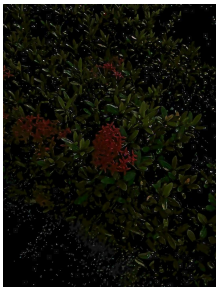
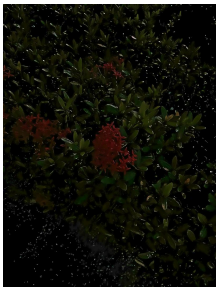
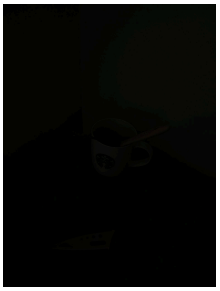
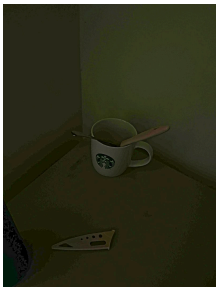
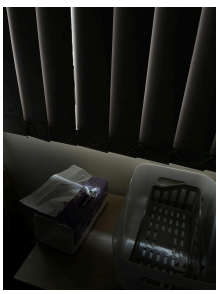
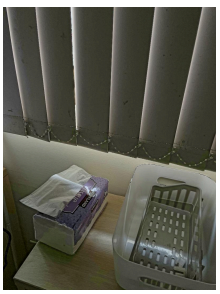
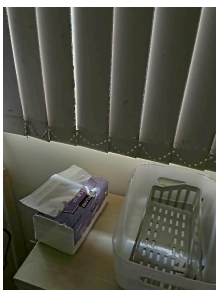
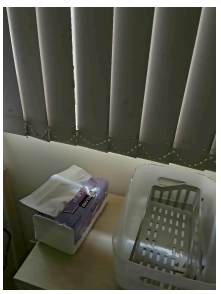
From the below table, we can see that the fast algorithm can in fact get similar results with the brute force method. The mean value of both PSNR and PIQE are very close. However, using this fast algorithm, we can save a lot of time. To see more detail in each image, the next table shows PIQE, PSNR, and time cost for each image.

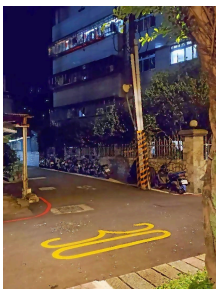
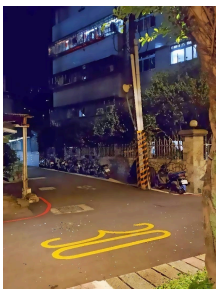
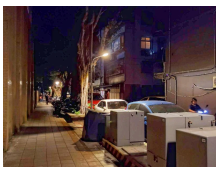
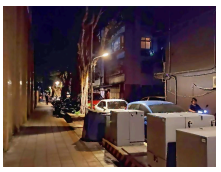
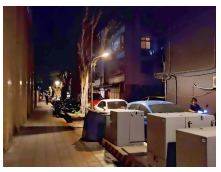
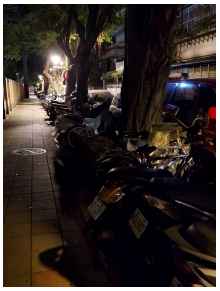
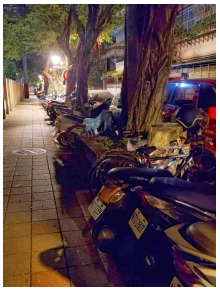
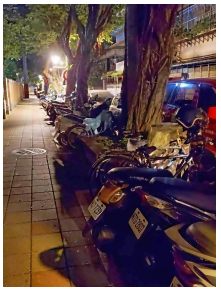
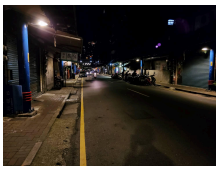
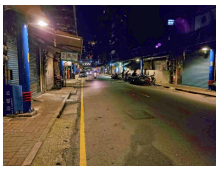
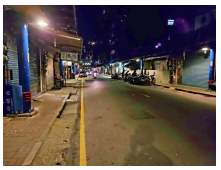
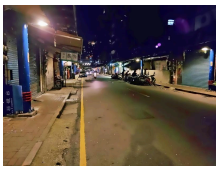
method	PSNR (mean)	PSNR (std)	PIQE (mean)	PIQE (std)	time (s) (mean)	time (s) (std)
fast	34.69	2.57	53.13	8.70	18.72	3.19
bruteforce	35.15	2.48	56.59	8.19	43.01	8.17

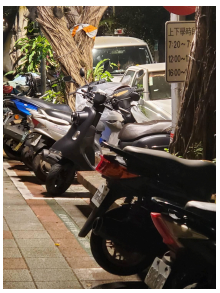
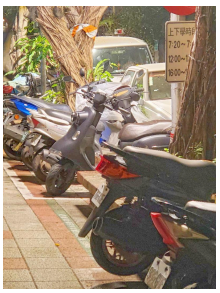
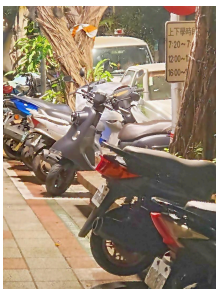
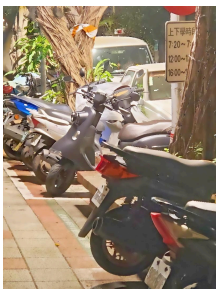
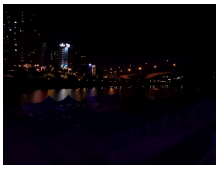
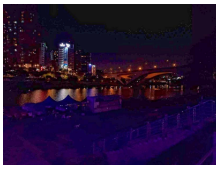
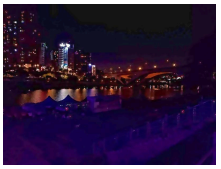
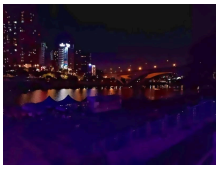
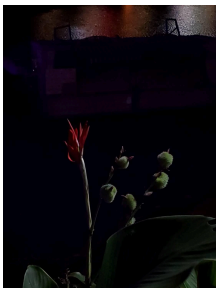
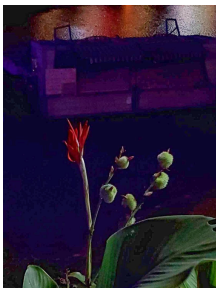
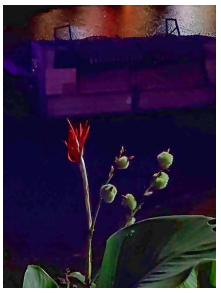
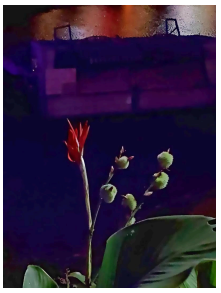
▲ Comparison between our fast algorithm and brute force method.

original	low light enhanced	fast	brute force
 1.jpg (**24.40**)	 1.29 s (27.59, 28.73)	 17.78 s (35.47, 61.52)	 44.33 s (**35.70**, 65.13)
 2.jpg (59.96)	 1.24 s (29.29, 51.00)	 18.91 s (32.12, 46.76)	 51.11 s (**32.60**, **44.76**)

 3.jpg (38.72)	 1.42 s (28.56, **26.77**)	 18.90 s (32.56, 37.69)	 49.25 s (**32.78**, 43.22)
 4.jpg (53.11)	 1.22 s (28.53, 49.82)	 20.36 s (32.95, **46.31**)	 48.84 s (**33.23**, 49.35)
 5.jpg (64.74)	 1.22 s (29.73, 56.20)	 18.65 s (32.91, **53.56**)	 44.24 s (**33.40**, 53.57)
 6.jpg (58.14)	 1.25 s (28.98, **40.69**)	 18.60 s (33.57, 55.46)	 44.16 s (**34.08**, 60.73)

 7.jpg (74.32)	 1.24 s (32.24, **64.38**)	 18.63 s (33.63, 64.87)	 44.00 s (**34.58**, 64.81)
 8.jpg (76.55)	 1.74 s (27.37, **70.52**)	 18.84 s (**42.33**, 70.69)	 43.93 s (42.28, 70.97)
 9.jpg (**51.73**)	 1.26 s (27.87, 53.17)	 19.76 s (38.76, 56.00)	 44.29 s (**38.83**, 57.41)
 10.jpg (34.41)	 1.26 s (27.58, **29.15**)	 18.45 s (36.09, 61.79)	 44.36 s (**37.20**, 64.72)

 11.jpg (43.62)	 1.27 s (27.75, **36.54**)	 18.56 s (33.66, 47.35)	 44.05 s (**34.09,** 51.67)
 12.jpg (44.53)	 1.27 s (27.77, **37.96**)	 18.79 s (33.48, 45.77)	 43.89 s (**33.98**, 49.92)
 13.jpg (62.87)	 1.13 s (27.69, **55.73**)	 10.38 s (34.43, 63.90)	 24.81 s (**35.21**, 70.01)
 14.jpg (38.94)	 1.27 s (27.68, **33.64**)	 19.02 s (33.09, 41.91)	 44.30 s (**33.51**, 48.99)
 15.jpg (41.77)	 0.79 s (27.63, **40.59**)	 10.42 s (33.62, 49.09)	 26.42 s (**34.01**, 53.46)

 16.jpg (37.54)	 1.24 s (27.59, **26.98**)	 19.02 s (32.15, 44.16)	 51.29 s (**32.73**, 49.39)
 17.jpg (62.56)	 0.78 s (28.27, 62.30)	 10.59 s (36.54, **54.72**)	 27.05 s (**37.15**, 59.22)
 18.jpg (**50.42**)	 1.24 s (28.12, 56.55)	 25.04 s (37.09, 54.74)	 53.90 s (**37.39**, 61.21)

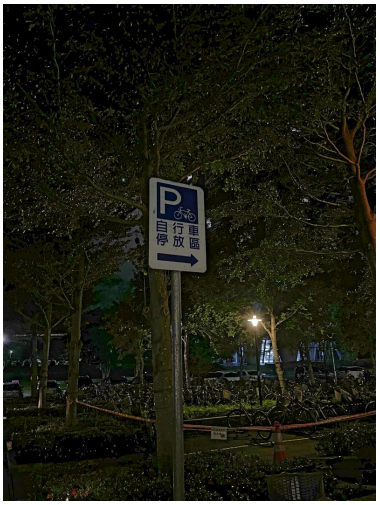
▲ fast and brute-force results after low light enhancement;
with the processing time and (PSNR, PIQE);
value in bold are the highest among the same image

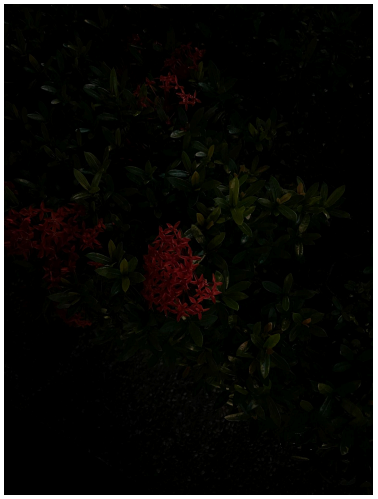
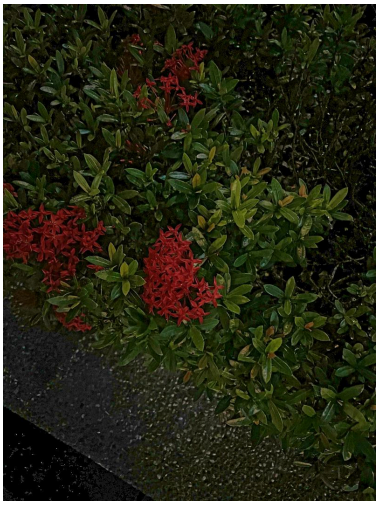
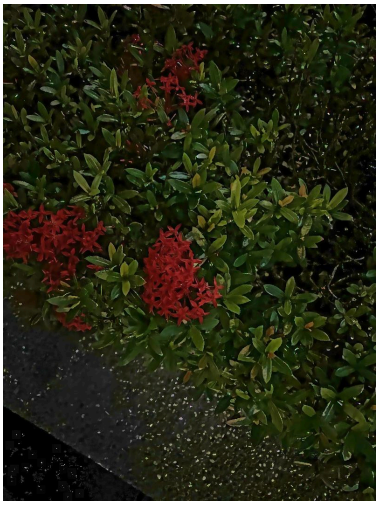
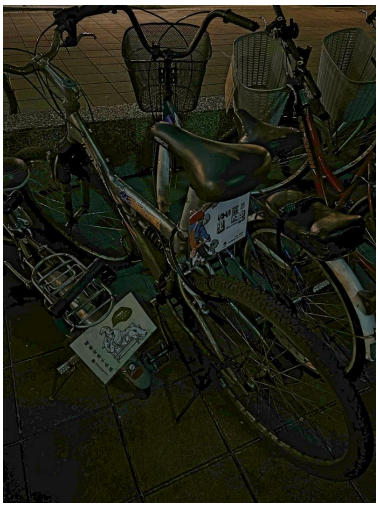
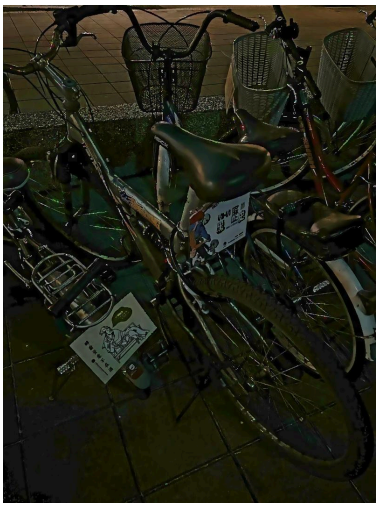
06. Combined result with adaptive bilateral filtering

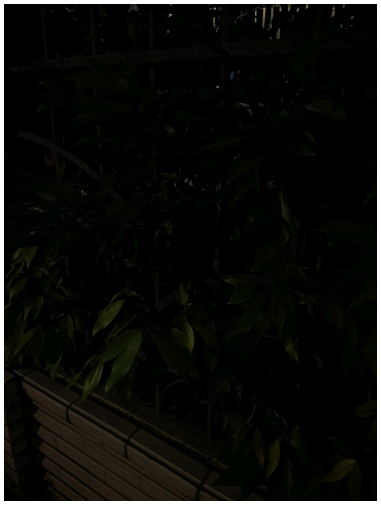
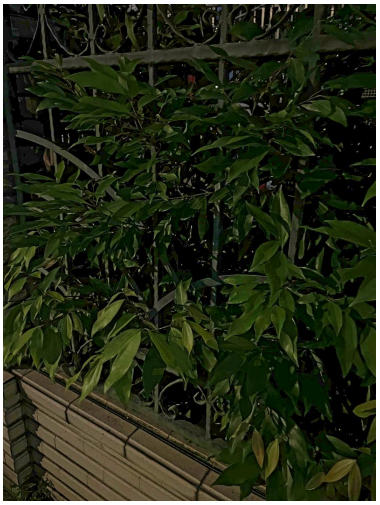
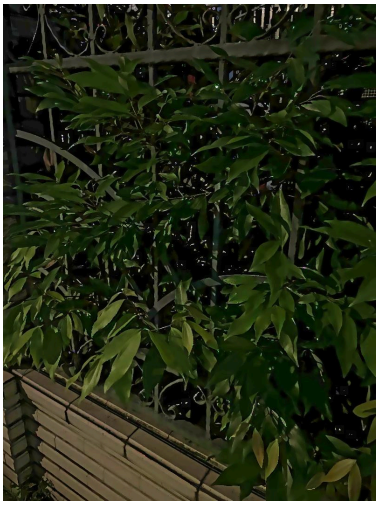
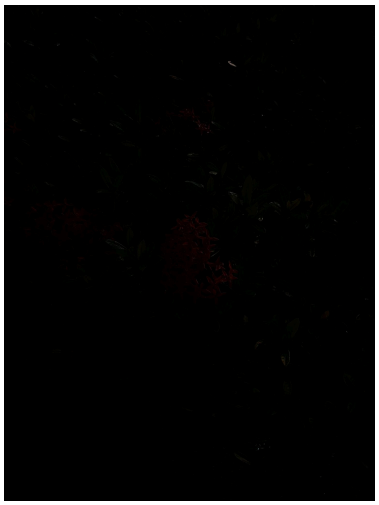
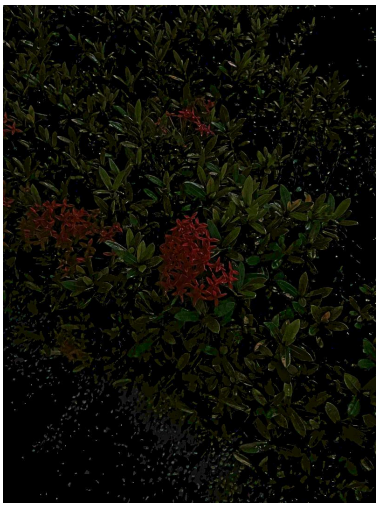
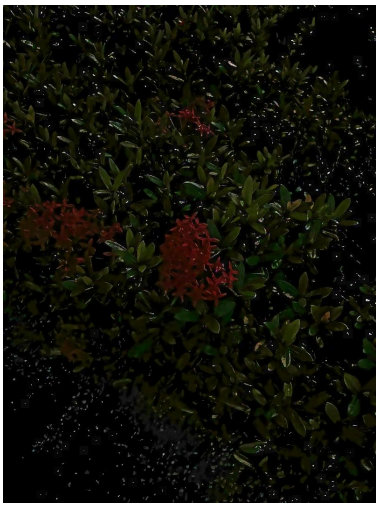
The below table indicates the processed results with our designed approach. We only processed the low light enhancement if our darkness check said it was needed. So, the “11”, “12”, “16” these three images were considered bright enough. This could be because the whole image has largely an area that is bright and clear. And we can see that no matter the image is taken outdoors or indoors, they all could have a recognizable enhancement.

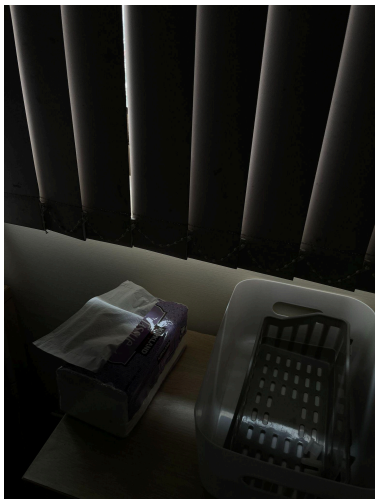
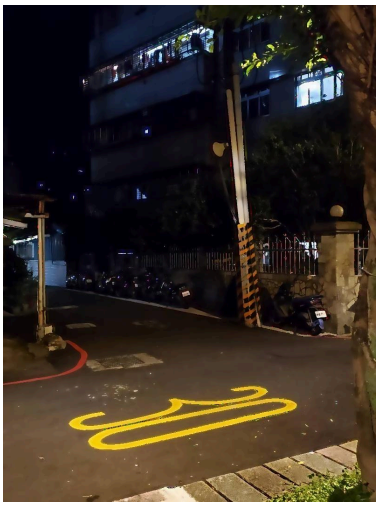
In most of our cases, we can get a better result after performing sharpening in the second step. However, sometimes the sharpened image may lower the PSNR or increase PIQE value. We can take 2.jpg for example. The output image sharpens leaves, but the image after low light enhancement already has a satisfying result. This causes the edges to become “too sharp”. In other words, the resulting images contain artifacts.

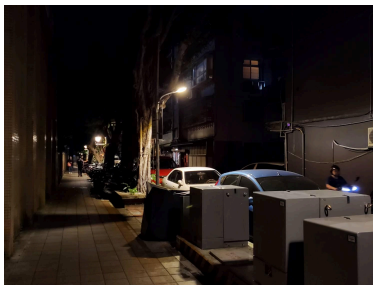
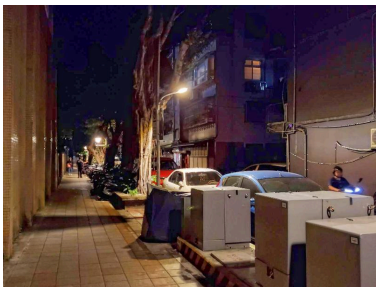
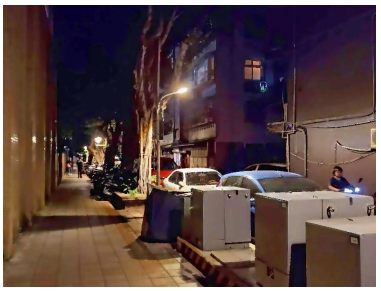
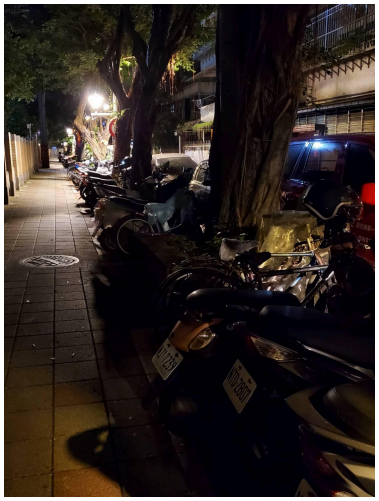
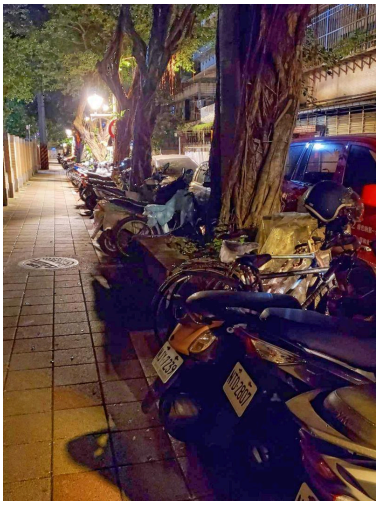
We can also take a look at 17.jpg. In the lower part of the enhanced image, the color of each pixel is too similar. As a result, after applying our algorithm, it treats it as the same texture, not edges. This causes the output image to become blurry in the below part of the image.

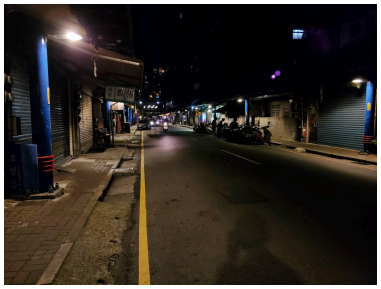
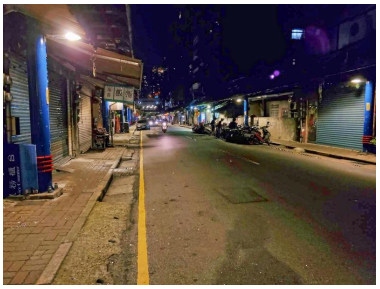
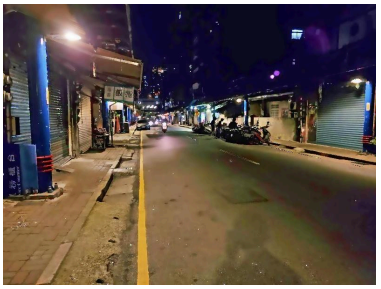
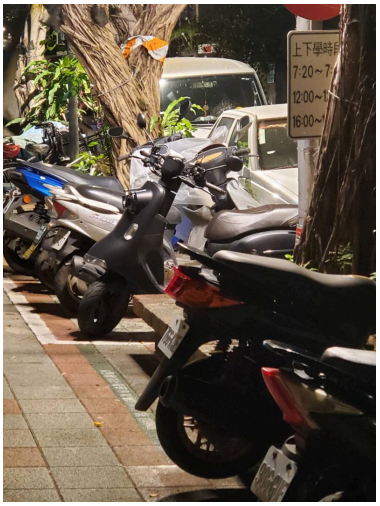
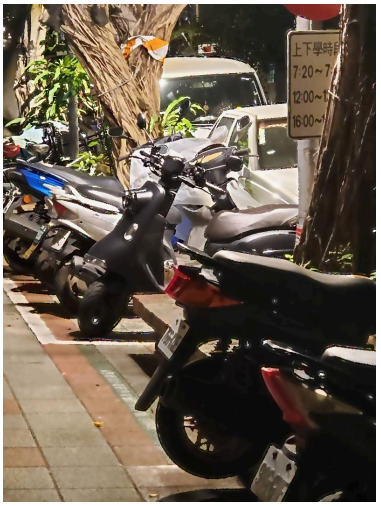
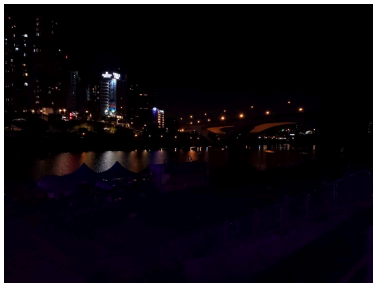
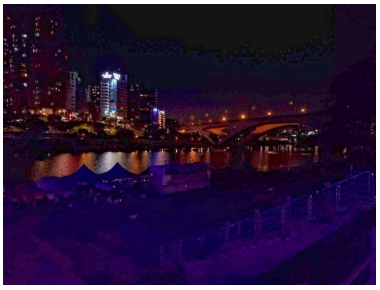
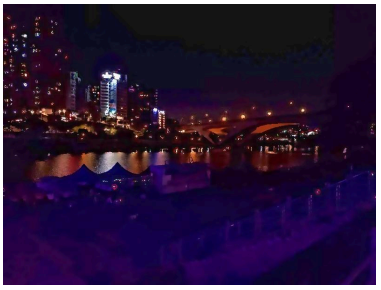
original	low light	combined
 1.jpg (24.40)	 1.29 s (27.59, 28.73)	 19.74 s (35.47, 61.52)
 2.jpg (59.96)	 1.24 s (29.29, 51.00)	 20.55 s (32.12, 46.76)

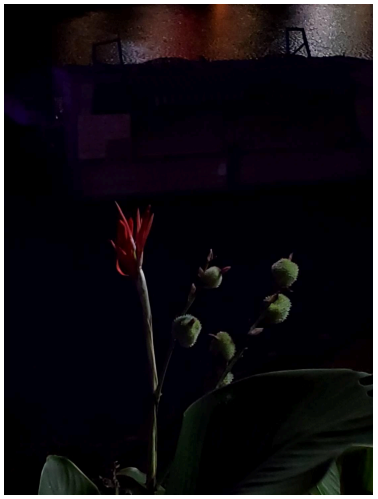
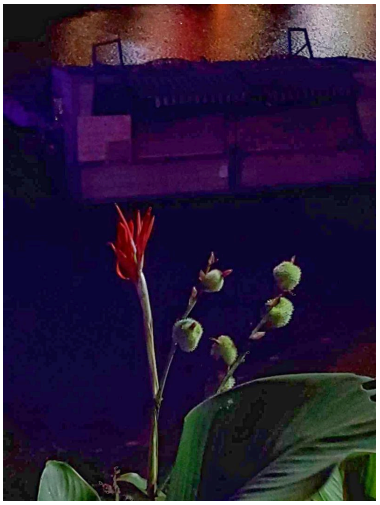
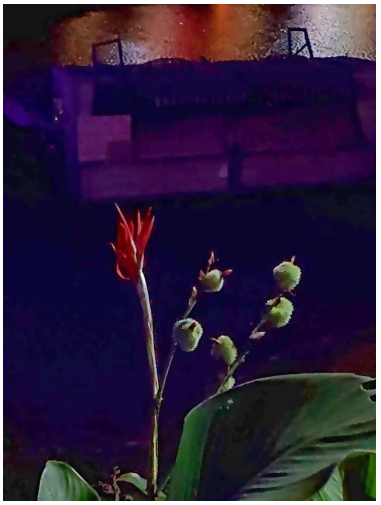
 3.jpg (38.72)	 1.42 s (28.56, 26.77)	 19.69 s (32.60, 37.69)
 4.jpg (53.11)	 1.22 s (28.53, 49.82)	 20.54 s (32.95, 46.31)
 5.jpg (64.74)	 1.22 s (29.73, 56.20)	 19.95 s (32.91, 53.56)

 6.jpg (58.14)	 1.25 s (28.98, 40.69)	 20.29 s (33.57, 55.45)
 7.jpg (74.32)	 1.24 s (32.24, 64.38)	 22.17 s (33.63, 64.87)
 8.jpg (76.55)	 1.74 s (27.37, 70.52)	 20.94 s (42.33, 70.69)

 9.jpg (51.73)	 1.26 s (27.87, 53.17)	 20.58 s (38.76, 56.00)
 10.jpg (34.41)	 1.26 s (27.58, 29.15)	 19.96 s (36.09, 61.79)
 11.jpg (43.62)	 not enhanced	 20.07 s (35.09, 57.13)

 12.jpg (44.53)	 not enhanced	 20.11 s (34.81, 58.67)
 13.jpg (62.87)	 1.13 s (27.69, 55.73)	 11.04 s (34.43, 63.90)
 14.jpg (38.94)	 1.27 s (27.68, 33.64)	 19.52 s (33.09, 41.91)

 15.jpg (41.77)	 0.79 s (27.63, 40.59)	 11.32 s (33.62, 49.09)
 16.jpg (37.54)	 not enhanced	 20.27 s (32.55, 54.34)
 17.jpg (62.56)	 0.78 s (28.27, 62.30)	 11.25 s (36.54, 54.72)

		
18.jpg (50.42)	1.24 s (28.12, 56.55)	19.34 s (37.09, 54.74)

▲ final results of our implementation with (PSNR, PIQE) and overall process time

VII. Reference

S. Ghosh and K. N. Chaudhury, “Fast bright-pass bilateral filtering for low-light enhancement”, Proc. IEEE International Conference on Image Processing (ICIP), pp. 205-209, Taipei, Taiwan, 2019

S. Ghosh, P. Nair, and K. N. Chaudhury, “Optimized Fourier bilateral filtering,” IEEE Signal Processing Letters, vol. 25, no. 10, pp. 1555-1559, 2018

R. G. Gavaskar and K. N. Chaudhury, "Fast Adaptive Bilateral Filtering of Color Images," *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019, doi: 10.1109/ICIP.2019.8802987

H. Ibrahim and N. S. Pik Kong, "Brightness Preserving Dynamic Histogram Equalization for Image Contrast Enhancement," in IEEE Transactions on Consumer Electronics, vol. 53, no. 4, pp. 1752-1758, Nov. 2007, doi: 10.1109/TCE.2007.4429280.keywords: {Brightness;Histograms;Image enhancement;Dynamic range;Consumer electronics;TV;Pixel;Digital images;Filters}

X. Guo, Y. Li and H. Ling, "LIME: Low-Light Image Enhancement via Illumination Map Estimation," in IEEE Transactions on Image Processing, vol. 26, no. 2, pp. 982-993, Feb. 2017, doi: 10.1109/TIP.2016.2639450.keywords: {Lighting;Estimation;Image enhancement;Visualization;Atmospheric modeling;Histograms;Image color analysis;Illumination estimation;illumination (light) transmission;low-light image enhancement}

C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271), Bombay, India, 1998, pp. 839-846, doi: 10.1109/ICCV.1998.710815. keywords: {Filtering;Color;Low pass filters;Photometry;Imaging phantoms;Pixel;Shape measurement;Smoothing methods;Computer science;Humans}