EE604A: Image Processing - Assignment 4

Piyush Kumar Gaurav (Roll No. 20104442)

November 14, 2021

1 Solution 1

The De-noising Technique used for Assignment 4 Question 1 "Any Filter: anyf" is **Median Filter**. It is a non-linear filter where each pixel is replaced by the median intensity value among all the neighbouring pixel of kernel size "N". The noisy pixel usually show abnormally different value of pixel intensity as compare to the values of neighbouring pixels. Median filtering thus replace noisy pixel values by more benign value (median). Median filtering was done for two images namely: "rome.jpg" and "iitk.jpg". Kernel size "N" was varied for (5×5), (7×7), (11×11), (13×13), (15×15). Following image shows the comparison:

Figure 1: This shows the comparison of rome image when undergone median filtering with different kernel size.

Figure 2: This shows the comparison of iitk image when undergone median filtering with different kernel size.

It was noticed that the noise in both the images (rome and iitk) has granular noise of random size. Since the size of the noisy pixel was not fixed it was difficult to remove all of them. Furthermore, when the kernel size was increased the blurrness increased leading to loss of edges. In order to show these observations precisely a zoomed portion of rome image was operated through same median filters

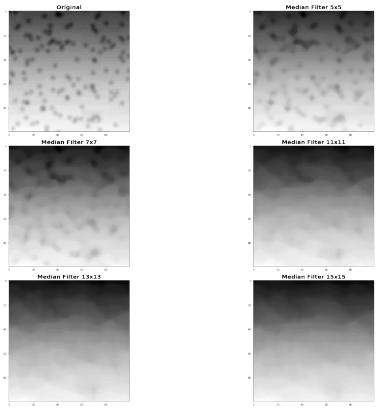

Figure 3: This shows the comparison of a zoomed portion of rome image when undergone median filtering with different kernel size.

Figure 4: This shows the output of Denoising using Median filter with kernel size = 7

2 Solution 2

In order to improve the speed in the implementation of bilateral filter, *mybf_original.py* it self was coded in an optimum way. Following are the improvements considered while implementing bilateral filter (point 1 implemented for both *mybf_original.py* and *my_BF..py* whereas other point was implemented only for *my_BF.py*)

1. Fix spatial kernel - Since spatial kernel is just a function of kernel size, it will remain fix for the entire image. Hence spatial kernel was computed and stored.

2. Image downscaling - The computational complexity of bilateral depends upon the number of pixels to be traversed during computation of intensity kernel and its convolution. Hence if we downscale the image and then compute the bilateral filter, traversal will happen for a lesser number of pixel. Here, a downscaling of 0.75 was found optimal such that the image do not looses perceptual appearance.

3 Solution 4

The bilateral filter algorithm is a computationally intensive de-noising technique which preserves the edges along with removal of noise. This is evident by the enormous time taken by the original bilateral filter. The original bilateral filter was used to extract the reference denoised image for both "rome.jpg" and "iitk.jpg". All other methods output were compared with this original raw image. A modest attempt was made to improve the speed of the original bilateral filter (refer Solution 2) by using primarily the technique of downsampling. Figure 5 and 5 shows the output of original bilateral filter

Figure 5: This shows the output of Denoising using original implementation of speed improved Bilateral filter for rome image

Figure 6: This shows the output of Denoising using original implementation of speed improved Bilateral filter for iitk image

Utilizing the method proposed by (YTA09) a real time bilateral filter was developed. This method instead of computing range and spatial filter pixel by pixel, constructs Principle Bilateral Filtered Image Components (PBFIC). These PBFICs are made using discrete values of intensity and original image. Finally the corresponding PBFICs are interpolated to derive approximation of Bilateral filter. This method shows drastic improvement in computational time (refer **??** and **??**) without affecting the image accuracy measures (PSNR and SSIM).

Figure 7: This shows the output of Denoising using implementation of real time Bilateral filter for rome image

Figure 8: This shows the output of Denoising using implementation of real time Bilateral filter for iitk image

Furthermore, Real time bilateral filter was utilized (with best parameter kernel size = 7, sigma spatial = 2 and sigma intensity = 50) to denoise the rome and iitk image. Next the images were compressed using JPEG 99, JPEG 95 and JPEG 90 methods. The compression also acts as a denoising technique thereby reducing the PSNR slightly.

Figure 9: This shows the output of Denoising using real time Bilateral filter for rome image alongwith JPEG compression 99

Figure 10: This shows the output of Denoising using real time Bilateral filter for iitk image alongwith JPEG compression 99

Figure 11: This shows the output of Denoising using real time Bilateral filter for rome image alongwith JPEG compression 95

Figure 12: This shows the output of Denoising using real time Bilateral filter for iitk image alongwith JPEG compression 95

Figure 13: This shows the output of Denoising using real time Bilateral filter for rome image alongwith JPEG compression 90

Figure 14: This shows the output of Denoising using real time Bilateral filter for iitk image alongwith JPEG compression 90

Following Table shows the analysis of filtering methods applied on "rome.jpg" and "IITK.jpg" respectively

Performance Analysis for rome.jpg			
Algorithm	Speed	PSNR	SSIM
anyf	0.268 sec	21.382	0.693
mybf (original)	298.538 sec	N/A	N/A
mybf (speed improved)	142.360 sec	26.25	0.86
rtbf (para-1: good accuracy)	4.983 sec	21.594	0.791
rtbf (para-2: good speed)	4.453 sec	22.812	0.775
rtbf (para-3: best para)	4.666 sec	22.610	0.788
rtbf (best para) (JPEG 99)	4.494 sec	19.554	0.565
rtbf (best para) (JPEG 95)	4.379 sec	19.547	0.564
rtbf (best para) (JPEG 90)	4.825 sec	19.541	0.563

Table 1: Performance Analysis for rome.jpg

Performance Analysis for iitk.jpg			
Algorithm	Speed	PSNR	SSIM
anyf	1.353 sec	32.041	0.913
mybf (original)	2057.435 sec	N/A	N/A
mybf (speed improved)	1059.210 sec	38.22	0.95
rtbf (para-1: good accuracy)	32.476 sec	30.898	0.939
rtbf (para-2: good speed)	28.778 sec	32.554	0.931
rtbf (para-3: best para)	29.339 sec	31.731	0.936
rtbf (best para) (JPEG 99)	32.364 sec	32.361	0.800
rtbf (best para) (JPEG 95)	31.397 sec	32.349	0.800
rtbf (best para) (JPEG 90)	31.682 sec	32.326	0.798

Table 2: Performance Analysis for iitk.jpg