EE569 Digital Image Processing

HOMEWORK #1

HOMEWORK – Introduction to Digital Image Processing

Issued: 13/01/2020 **Due: 27/01/2020**

Problem 1: Image Demosaicing and Histogram Manipulation (50%)

(a) Bilnear Demosaicing (10%)

Motivation

The aim is to practice demosaicing method that can transform gray-scaled image to colorful RGB image. The bilinear transformation method is the simplest way to transform the gray-scaled image into the RGB image. The color type of three channels of the pixel depends on the position of processed pixel and its neighboring pixels. The formula to compute the pixel color value varies when position of the pixel changes. Hence, if-else argument is designed to compute pixel value with different position in the image to judge the color type of pixel.
The formula shows the key computation procedure for each pixel intensity value.

$$F(i,j) = \frac{1}{\#\ of\ neighboring\ pixels(2\ or\ 4)} * \sum_{n1,n2} F(i-n1, j-n2)$$

Each pixel intensity value of certain color is estimated by the neighboring pixels with the same color type. The pixel intensity value in three channels is computed with the same procedure because the edge is extended to avoid different solution for intensity value estimation.

Approach and procedure

This is the first problem to be solved. Hence, the basic C++ program method like inputting file, allocating memory for array to process problem generally will be introduced firstly. The procedure for following problem below will be explained again. The steps for setup are almost same for each problem.
Before starting computing, memory is allocated to store the initial image. To prevent different manipulation on the pixels on the edge, the original image is extended with mirror rule. The implementation of mirror rule is to mirror pixel with the first column. For example, the intensity value of pixels in the column at extended edge neighboring the first column of original image has the same value with the first column pixels. In that case, all pixels at the extended edge are mirrored by the first pixel column. The function *extend2DImage()* has finished this function.
The followings are the concise introduction to the general function:
read2DImageFile()
IO function to read image data to 2-dimensional array
write2DImageFile()
IO function to write processed image data to a file
extend2DImageEdge()
extend image edge to ensure the same operation for the pixels. The extension rule uses the first column as the mirror column for the closest. When the pixels is looped to solve some problems, the index of pixel is not the original pixel. For row and column index, the pixel of extended image can be accessed by the original row and column index in addition to the size of the edge. There are also functions to handle 3-D image with same technique.
alloc2DImage()
allocate memory dynamically to the 2D array to store the image
The RGB image pixel color value is computed by averaging the pixel value of same color type. Specifically, for example, it is different from red and blue for green pixel because there are only two neighboring green type.

Results

The result is amazing. The obtained image is colorful. The computation cost is low. There are O(n^2) complexity for the whole computation procedure. The Figure 1.1.1 shows obtained image computed by the bilinear algorithm from gray-scaled image. The Figure 1.1.2 shows the original image. With comparison to the original image, the obtained image is more colorful in some area. The grass is more green. In Figure 1.1.3, when the camera is zoomed, the distribution of color pixel intensity value of obtained image is more equal than the distribution of original image.

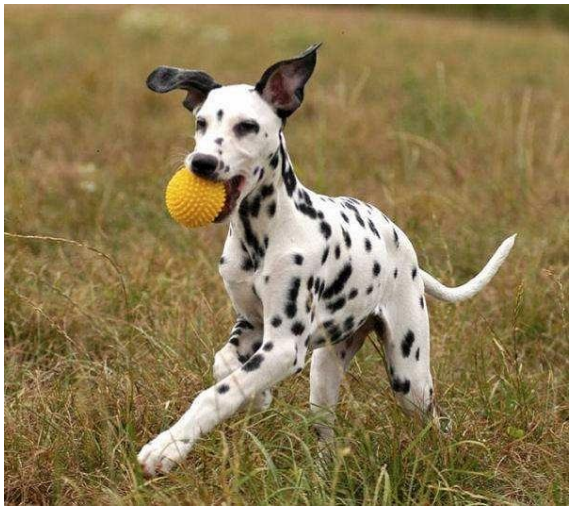

Figure 1.1.1 obtained image

Figure 1.1.2 original image

Figure 1.1.3 colorful artifact compared with the original image

Discussion

The computation of the pixels depends on the position of the pixel. The averaging process is safe. There is overflow for the final computed pixel. The maximum pixel value is 255. The addition for C++ is 2 byte operator space. The final result for each pixel will be within 0-255 because the total summation will be divided by 2 or 4, so there is no overflow. As the color distribution is equal for some area, the demosaicing procedure for bilinear algorithm loses some information.

Answers

1. In Figure 1.1.1 and Figure 1.1.2, you are shown with the obtained dog picture and the original dog image. Compared with the original image, the obtained image is more colorful. The grass is much greener. More shapes can be viewed from scanning. Some information is lost.

2. Problem may be caused by the local average process for computation. The bilinear method ignores the global hue for the image, the color seems to be distributed equally from the obtained image.

(b) Malvar-He-Cutler (MHC) Demosaicing (20%)

Motivations

The MHC method to demosaice image is to add small computation compliment when we compute color value of three channels for the image.

Approach and procedure

1. Judge the primary color of pixel.
2. Compute difference between value of original color and neighbor color.
3. Compute estimated value for other color of the referenced pixel.
4. Loop the above three steps for all pixels and get the image.

The computation procedure for difference value between original color and neighbor color depends on the color category of the pixel, so the specific function is chosen to compute the delta value for the pixel. There may be overflow for the unsigned char data type, so a compliment function needs to set threshold for each pixel intensity value.

Results

Figure 1.2.1 shows an image with black background. Some area of the obtained image contains much more colorful color that is stronger than the original image.

Figure 1.2.1 Obtained Image with MHC method

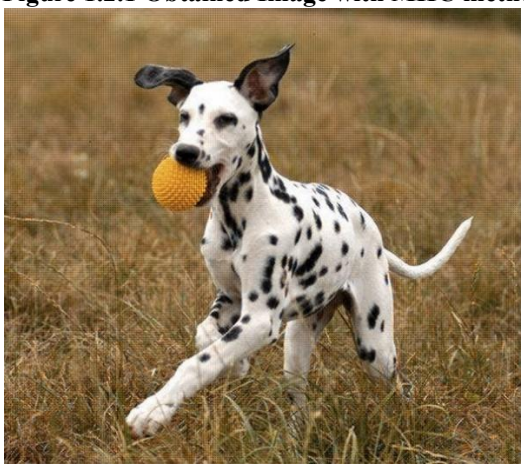

Figure 1.2.2 original image

Discussion

Overflow happens when the increment compliment estimation by the neighboring pixels is computed. Hence, the negative value may appear. The possible result can be negative for the pixel value. This cause the overflow for the unsigned char type because the unsigned char value cannot be negative. The uncorrelated value causes several peak pixel with pure red, blue that is not correlated to the neighboring pixel value. The solution to eliminate this noise is to use threshold to restrict the range of the pixel to 0 to 255.

Answers

1. The figure above gives the clear comparison between the original image and image processed by Bilinear Demosaicing algorithm
2. The artifacts are the image that are more colorful than the other image processing.

c). Histogram manipulation(10%)

Motivation

The aim of histogram manipulation is change the distribution of pixel intensity value. The foundation method to implement it is to construct a transformation function that maps the pixels for the same value to another pixel value. The problem provides two principle for us to construct transformation function for the pixel value mapping. The method A is to use cumulative probability theory. The probability density for each pixel value should be uniformed so that the probability distribution can be transformed to uniform distribution. [1].

Approach and procedure

Method A transfer function based histogram equalization

1. compute the number of pixels corresponding to dedicated one gray-scaled value
2. get the histogram to describe the distribution between pixels numbers and gray-scaled intensity value
3. sum from the histogram value of pixel with 0 intensity value to the wanted histogram value and then give the sum histogram value for each pixel intensity value.
4. apply normalized transfer function based on the sum histogram to transfer current image pixels with certain intensity value to other intensity value.

TransferFunctionBasedHistogramEqualization()
Construct transfer array to map the original pixels to the equalized pixels
histogramCountByChannel()
count the number of pixels with the same intensity value, each channel is counted separately
writeHistogramArray()
write histogram array to the file for convenient plot by matlab
poltTransformArray.m
plot the transform array for each channel by matlab

Method B cumulative probability based histogram equalization

1. rearrange each pixel position. The pixel with the same intensity value is positioned in the section of array.
2. row and column index in the image of pixel and intensity value of pixel is separately recorded in the array.
3. The pixel value is reallocated from 0 to 255 sequentially.
4. Extract the data from the reordered array and put them into 2 dimensional array for image output

RandomPickHistogramEqualization()
RandomPickBasedHistogramEqualizationByChannel()

Results

Figure 1.3.1 shows the histogram distribution of the original image. Figure 1.3.2 shows the histogram of the image processed by method A. From comparison between two figures, the pixels value distribution is transferred more equally. The contrast of image is improved from the comparison between Figure 1.3.5 and Figure 1.3.6. The transfer function itself is almost a linear function that maps pixels of original image to the pixels processed by the method A. In Figure 1.3.4, the values in the histogram of three channels are the same. The method B transfers the probability distribution of pixels value to the uniform distribution. Both methods are quite effective to transfer the original image to the image which has more contrast.

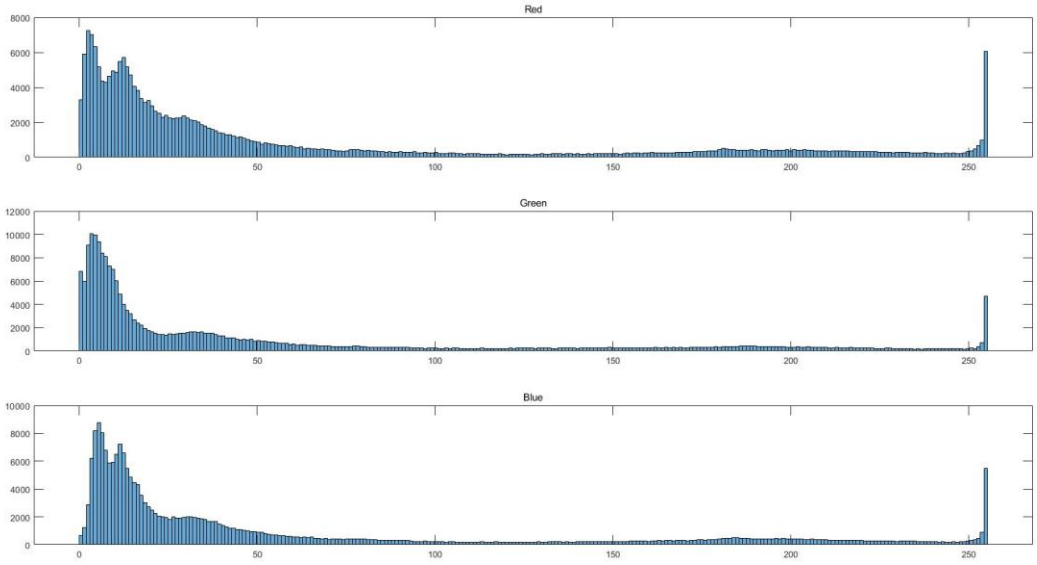

Figure1.3.1 histogram of original Toy.raw image

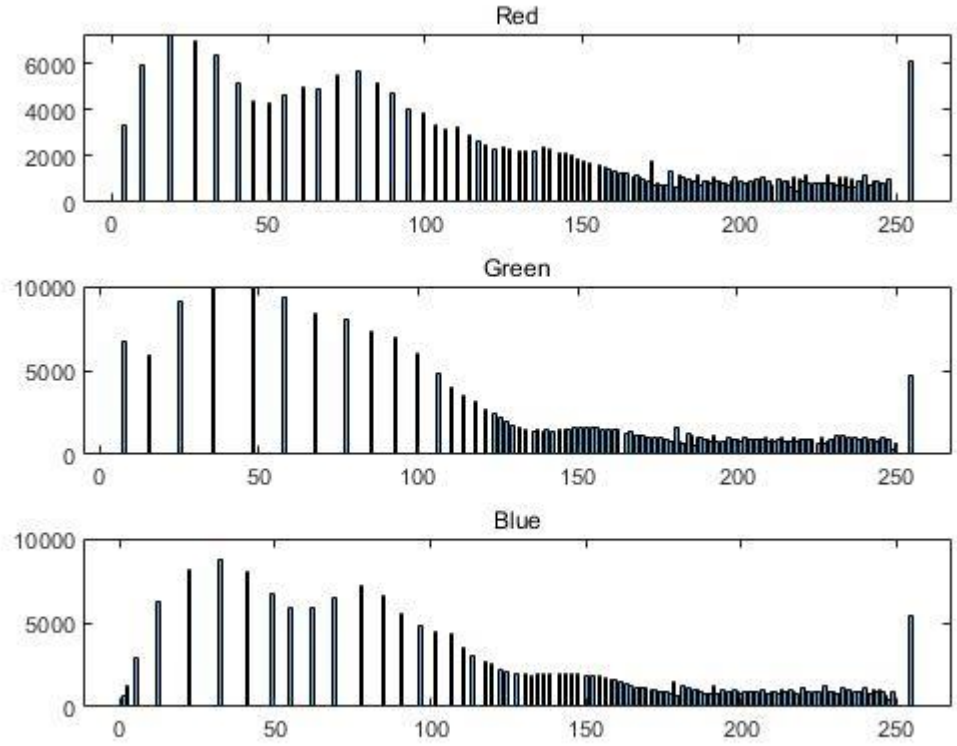

Figure 1.3.2 histogram of obtained Toy_a.raw by method A

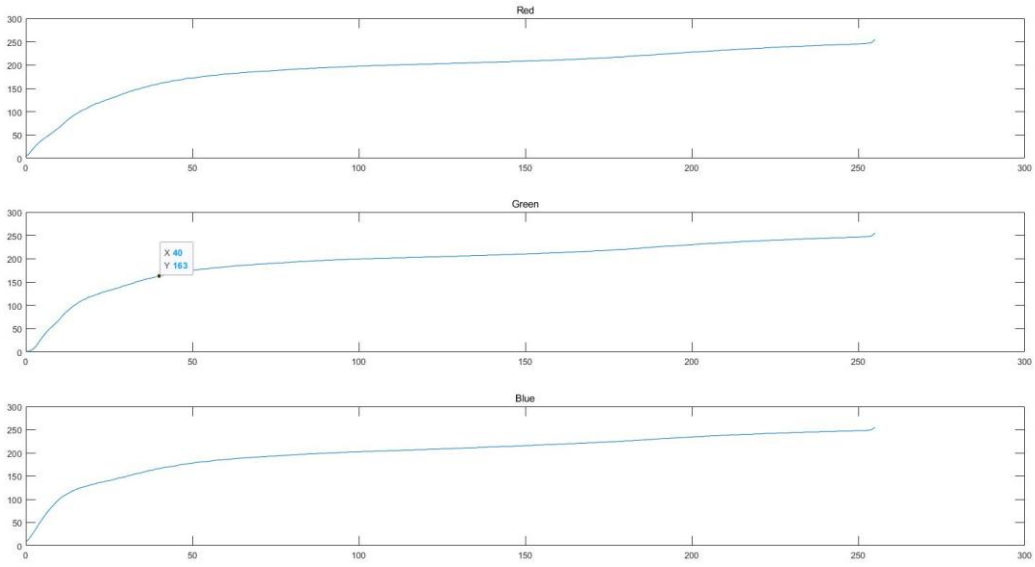

Figure 1.3.3 mapping transfer function

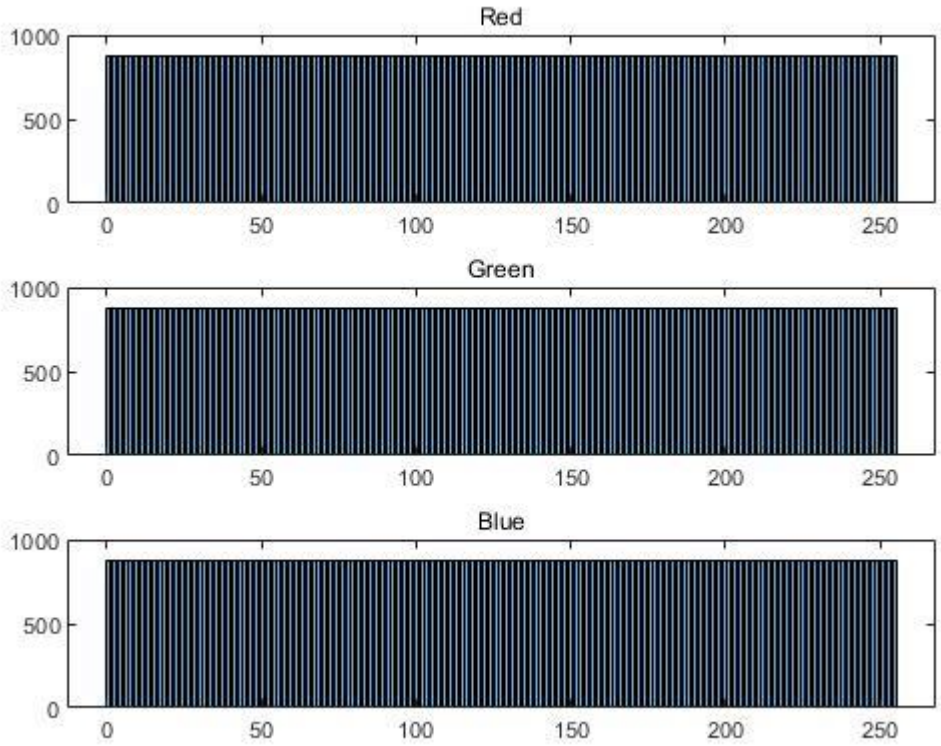

Figure 1.3.4 histogram of obtained Toy_b.raw by method B

Figure 1.3.5 the original toy image

Figure 1.3.6 obtained toy image with method A

Figure 1.3.7 obtained Toy_b.raw image with method B

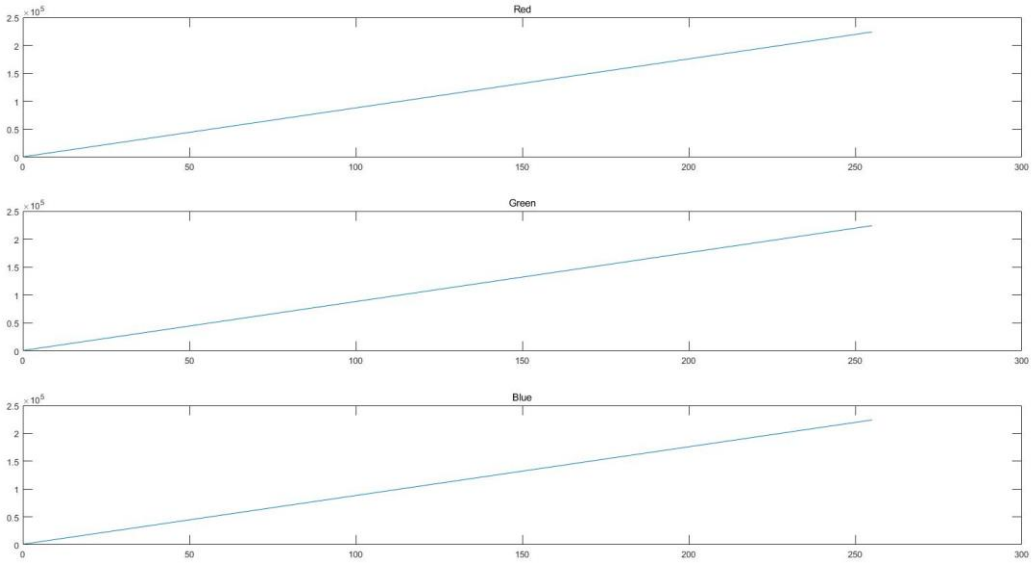

Figure 1.3.8 cumulative histogram for channels transferred by method B

Discussion

The first try to get RGB image does not consider the boundary problem, some overflow may appear because the intensity value result is much bigger than 255 or much smaller than 0. The type that is used to store each pixel value is unsigned char which is only allocated one byte. The compliment method is to give pixel value 255 or 0 when big overflow or negative overflow happens.

Answers

1. The figure 1.3.1 shows the histogram for the original image. Three channels are separately described by one histogram distribution table.
2. Figure 1.3.3 plots the transfer function of method A
3. Figure 1.3.8 plots cumulative histogram of method B
4. The histogram of original image shows there are much pixels concentrated in around 0 and 255, quite white and black pixels are much more. This concentration on both black and white sides causes contrast of image is not sufficient to show the details of some area. The histogram equalization is to redistribute the histogram distribution of pixel value and make contrast of different region in image show more detailed information

which can be clearly seen by naked eye. Both of two histogram equalization method gives good result. The difference is not obvious for naked-eye result. The histogram result shows the difference of two method. For method A, the histogram distributed more equally but not precisely uniform. And the mapping function of method A is approximately linear. The histogram distribution of method B is precisely uniform. The cumulative histogram is almost linear. These two methods treats problem within different metric space

Problem 2: Image Denoising (50%)

(a) Basic denoising methods (10%)

Motivation

This is the simplest method among the problems to denoise image. Each pixel are obtained by averaging the neighboring pixels with the same weight. This weighs neighboring pixels equally by default. This assumption is defective when there are some discontinuous changes in the image.

$$pixel\ value = \frac{\sum_{all\ pixels} f(i,j) * I(i,j)}{\sum_{all\ pixels} f(i,j)}$$

Approach and procedure

1. Do position offset for each pixel to compliment the edge extension offset
2. Averaging the referenced pixel with neighboring pixel in the window
3. Loop all pixels repeating the above steps and get the image

linear_filter()
loop all pixels to estimate them with uniform weight filter
aver2DImage()
compute intensity value of pixel by averaging all neighboring pixels with uniform weight
GaussianFilter()
Loop all pixels to estimate them with gaussian weight filter
compGaussianPixel()
compute intensity value of pixel by averaging all neighboring pixels with gaussian weight

Results

Figure 2.1.1 shows the original pure image and the denoised image by bilinear. Figure 2.1.2 shows a set of images whose window size is tuned large. The sets of images show that when the window size is more large, the processed image is more obscure. The PSNR is lower when the window size is much larger. The trader-off should be placed on the final tuning parameters. Table 2.1.1 shows the debug parameter for the image denoising. The PSNR is computed by comparison between the original image and the denoised image. The window size is firstly tuned to improve performance. After window size gets the value where PSNR gives good performance for one certain filter type, then the type of filter is changed for comparison. From the table, we can see the gaussian filter performs well when the window size is large. The uniform filter works bad when the window size is very large. By contrast, the PSNR can still keep satisfying result when window size is 20 for gaussian filter.

Figure 2.1.1 the denoised image with method A and the original image

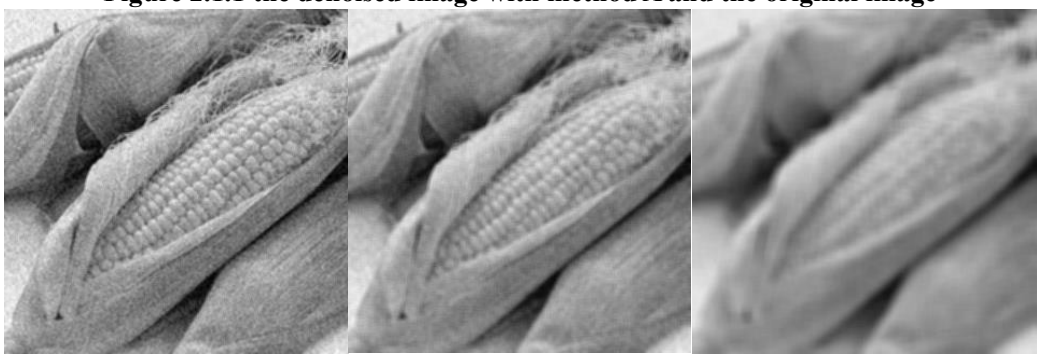

Figure 2.1.2 the denoised image set by uniform filter

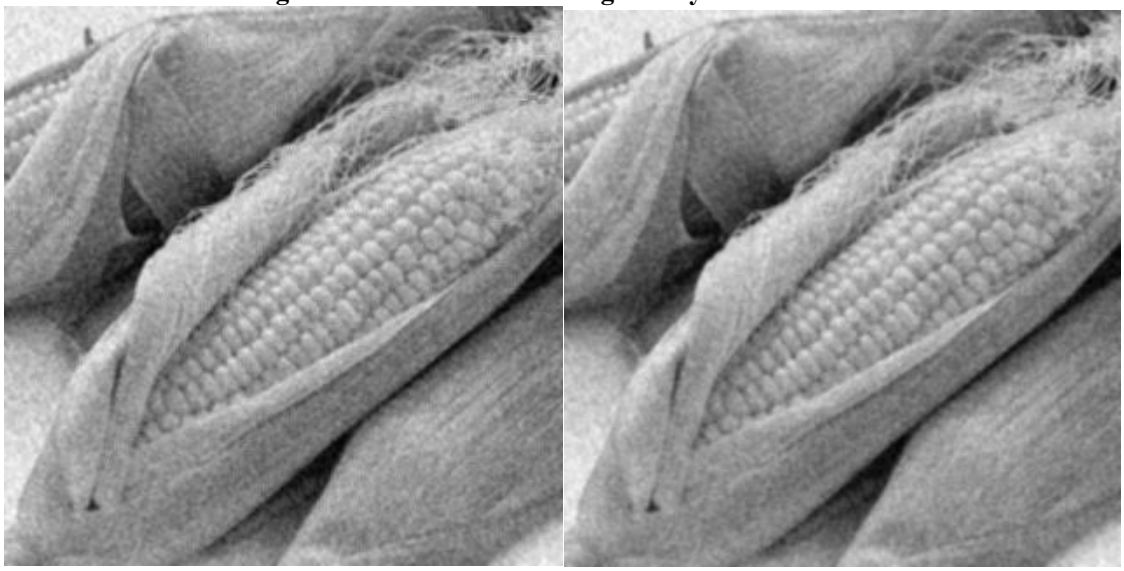

Figure 2.1.3 the denoised image set by Gaussian filter

ID	Filter type	Window size	σ	PSNR	comment
1	uniform	4	n/a	19.0771	

2	uniform	6	N/A	18.8386	
3	uniform	8	N/A	18.6364	
4	gaussian	10	N/A	18.4677	
5	gaussian	4	1	19.3856	
6	gaussian	6	1	19.3827	
7	gaussian	10	1	19.3825	
8	gaussian	20	1	19.3825	
9	gaussian	10	10	18.5714	
10	gaussian	4	5	19.0911	

Table 2.1.1 debug parameters for uniform and gaussian filter

Discussion

The window size can be tuned to make the algorithm perform better. The PSNR is generated by program and when PSNR is high, the performance of algorithm is much better. Table 2.1.1 shows that only small window size can make both of algorithms work well. When the window size is tuned larger, which means more pixels are considered as neighboring pixels in the window, the gaussian filter is less influenced by enlargement of the window size. As the window size become larger, the performance gets worse for both filter. The decline of PSNR is caused by the addition of much noisy neighboring pixels.

Answers

1. The image contains gaussian noise.
2. The difference between filter of uniform weight function and gaussian weight function is the window size tuning. The filter of uniform weight performs worse when the window size is large. In contrast, there are only slight effect on PSNR performance for gaussian filter. The reason is that the gaussian weight function take the position of neighboring pixel into consideration to compute weight. The weight will be very small for large distance from the index position of the processed pixel. There is another parameter for gaussian filter. In table 2.1.1, it works well when parameter is 1. There is no improvement when σ is tuned larger than 1.

(b) Bilateral Filtering (10%)

Motivation

Bilateral filter compute weight based on Gaussian Probability distribution. It measures the pixels and neighboring pixels by their relative position and intensity pixel value relationship. Both effects on the pixel relationship decides the gaussian coefficients for the pixel intensity value. When the distance between measured pixel and target pixel is large, the weight for the measured pixel will be small. The extension of effect of both elements on the algorithm performance is based on the parameter of standard variance that can be tuned.

$$weight = exp(-\frac{relative\ position\ distance^2}{2\sigma_c^2} - \frac{difference\ of\ pixel\ intensity\ value^2}{2\sigma_s^2})$$

Approach and procedure

1. Loop for each pixel to compute denoised intensity value
2. Compute the weight for each pixel in the window
3. Sum up and average the pixels in the window to get the intensity value

bilateral_filtering()

loop all pixels to get the estimated results from *computeBilateralFilteredPixel()*

computeBilateralFilteredPixel()

compute pixel intensity value by summing up all gaussian coefficients multiplied by the pixel intensity value and normalizing by summing up all gaussian coefficients themselves

computeGaussWeight()

compute the gaussian function result by the relative distance between processed pixel and the selected pixel in the window loop procedure.

The more detailed procedure can be checked by scanning the program code.

Results

In Figure 2.2.1 shows the comparison of denoised image with different standard variance. At first glance, the right side image performs well in denoising. Figure 2.2.2 shows the comparison between denoised image with different window size. The right side is the denoised image with large window size. Hence, the large window size for the bilateral algorithm has benefits on improvement of performance. Table 2.2.1 summarizes the PSNR change trends as some parameters change. The Figures present different comparison between these parameters.

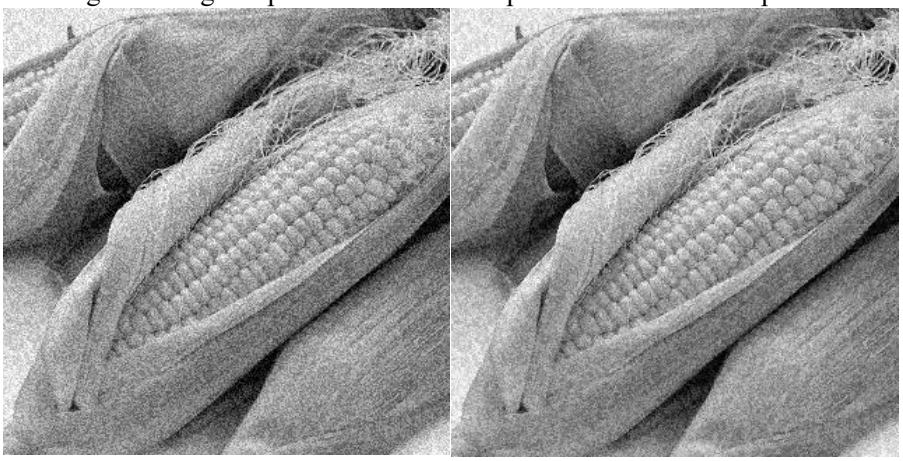

Figure 2.2.1

Figure 2.2.2

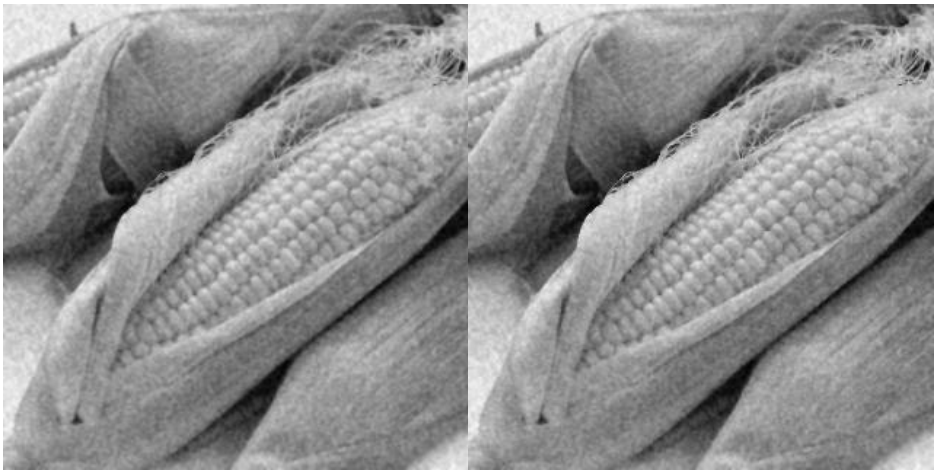

Figure 2.2.3

ID	window size	σ_C	σ_S	PSNR	Citations
1	4	1	10	17.7505	
2	4	1	20	18.3536	Figure 2.2.1
3	4	1	30	18.9423	Figure 2.2.1
4	4	1	50	19.4444	
5	4	1	100	19.5138	Figure 2.2.2
6	4	1	120	19.4876	
7	4	1	150	19.4552	
8	10	1	100	19.5159	
9	20	1	100	19.5159	Figure 2.2.2
10	10	5	100	19.0669	Figure 2.2.3
11	10	2	100	19.3528	Figure 2.2.3

Table 2.2.1 the debug parameter for bilateral filter

Discussion

As Table 2.2.1 shows, the perfect performance is given when the window size is around 10. There is no sharp improvement when window size continues to enlarge because gaussian coefficient declines sharply with the enlargement of distance between pixels. σ_c describes the distance from the processed pixel and neighboring pixel. Hence, the its value is around 1 as the pixel from far distance plays has slightly effect on the processed pixel value. σ_s describes the difference between processed pixel intensity value and neighboring pixels value. The range of these pixel values is from 0 to 255. It is reasonable that the performance is perfect when σ_s is around 100.

Answers

1. the figures and program show the result.
2. The σ_c and σ_s are different. σ_c describes the variance of relative position index. The better performance occurs at around 1. The σ_s describes the variance of pixel intensity value. The better performance occurs at around 75.
3. It has almost same performance with the algorithm in problem2(a). It has better performance if the parameter is tuned well, but the performance is just slightly higher. The highest PSNR is 19.51 which is slightly bigger than highest PSNR of filter in problem a.

(c) Non-Local Means(NLM) Filtering(10%)

Motivation

The basic idea of NLM algorithm is to "build a pointwise estimate of the image where each pixel is obtained as a weighted average of pixels centered at regions that are similar to the region centered at the estimated pixel" [2]. The size of window for the processing pixel to compute the distance from the neighboring pixel and the window to compute the gaussian coefficients is different. The implementation for the algorithm is complex. The code is separated into several parts to improve the debug efficiency. All hyperparameter can be adjusted by argument for program input. The formula is complex and hard to describe. The basic principle of the algorithm is to compute weight based on the patch of referenced pixel and neighboring pixel in the window by Euclidean distance. The detailed implementation can be checked in the given code.

Approach and procedure

1. Compute image pixel value in the window size, make relative edge size addition for each pixel position index
2. For each pixel value computation, the patch window is constructed and loops the pixels in the patch window.
3. Compute the Euclidean distance based on the looping of pixels in the patch window between patch of processed pixel and neighboring pixel.
4. Transform the Euclidean distance to the gaussian coefficient for each neighboring pixel
5. Estimate pixel value
6. Loop for all pixels

NLM_filtering()
Loop for all pixels and do the offset for extension of edge
computeNLMPixel()
estimate pixel value with neighbouring pixels in the window, weighted averaging all pixel value in the window
compEuclidianDistanceWeight()
transform area Euclidean distance between two pixels to the gaussian weight
compEuclidianDistanceArea2Area()
compute Euclidean Distance between two patches of two pixels

Results

Figures show the comparison between denoised image with different denoising parameter. All comparison is based on the principle that the number of different parameters is only one. This ensures the relation between parameters and denoising performance is strong. Table shows the detailed data and PSNR performance. The table gives the specific parameter values and PSNR performance for further research on parameters tuning.

Figure 2.3.1

Figure 2.3.2

Figure 2.3.3

Figure 2.3.4

Id	window size	patch size	Parameter h	Parameter σ	PSNR	Comment
1	6	4	10	10	19.3719	Figure 2.3.1
2	6	4	1	10	17.6887	Figure 2.3.1
3	6	4	20	10	19.1243	Figure 2.3.4
4	6	4	10	20	19.2585	
5	6	4	10	50	19.093	Figure 2.3.2
6	6	4	10	25	19.2143	
7	10	4	10	10	19.1479	Figure 2.3.2
8	6	4	5	10	18.6428	Figure 2.3.4
9	6	4	5	25	19.3354	
10	6	4	5	50	19.3469	Figure 2.3.3
11	6	4	5	100	19.2152	
12	5	5	10	10	19.4315	
13	10	5	10	10	19.1479	
14	10	5	6	10	19.2150	
15	4	4	6	20	19.3054	
16	4	4	10	20	19.4309	
17	4	4	8	20	19.455	
18	4	4	7	30	19.4552	
19	6	4	7	30	19.3421	
20	6	4	7	50	19.4064	
21	6	4	7	75	19.3485	
22	6	4	6	50	19.4428	
23	10	8	6	50	19.1711	
24	10	8	10	50	18.8339	
25	10	8	5	50	19.2639	
26	10	8	1	25	17.6949	
27	10	8	4	50	18.2249	
28	10	8	7	50	19.058	

Table 2.3.1

Discussion

Table 2.3.1 shows the debug result. The PSNR evaluate the performance of denoising algorithm. The result is the same when window size is large and other parameters are given by different sets of composition. From the trails from 20 to 28, it shows that the algorithm can endure the large window size. The NLM algorithm can get more global information but it costs much computation resources. The NLM algorithm can solve more general problem by tuning the window size and patch size. The h parameter presents the relative position between pixels in the filtering window. The standard variance is to measure the variance between pixel value. Both parameters should be tuned under their own boundary. For relative position, the range to tune the parameter is around 5. For standard variance, the range should be controlled around 25.

Answers

1. The h parameters for the initial weight for neighboring pixels decide the influence of patch size and standard variance parameter influence extension. When the parameter h is small like 5, it makes higher standard variance like 50 improve the result. The h parameter is to adjust the influence of neighboring pixels based on the distance. The table shows
2. The NLM algorithm performs bad compared with the algorithm above. It is hard to get good performance by tuning parameters. The running time for algorithm is much longer than time of algorithms above. The difference between NLM and other algorithm is that more global information is included in the algorithm. The algorithm can endure large window size but it needs long time to tune parameter. From my personal perspective, I will not recommend this algorithm for first trial. If there is other optional choice, I will never use it.

(d) Block matching and 3-D transform filter (10%)

Motivation

The BM3D algorithm is the most complex algorithm among the above denoising algorithm. This algorithm can be used in different complex scenario without careful analysis of image noise characteristics before. Hence, the algorithm performs well even if there is no tuning for the variance parameter. The performance is the highest among the algorithm above.

Approach and procedure

It is hard to implement the algorithm. The open source matlab code [3] is used to implement algorithm. When the path of image file for input and output is added, all procedure is finished.

Results

Figure 2.4.1 and Figure 2.4.2 shows comparison between two images. The performance of BM3D is quite good. The algorithm works efficient without much effort to tune the parameter and the performance is quite good even at the first trial.

Figure 2.4.1 the denoised image and PSNR value by BM3D algorithm

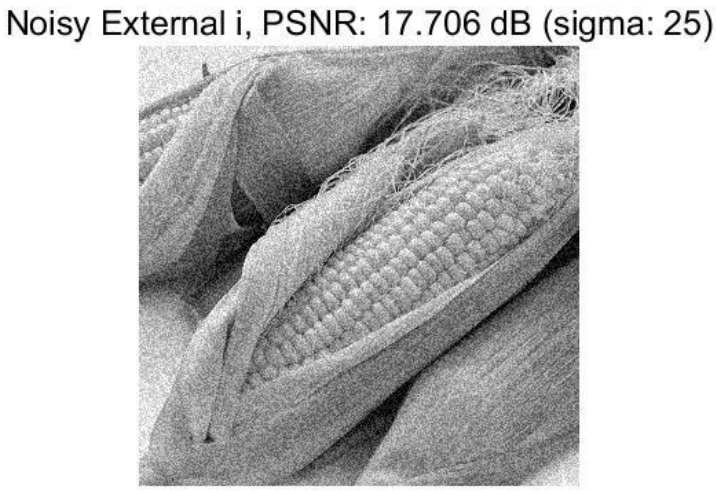

Figure 2.4.2 the original noisy image and the PSNR with the pure image

Discussion

The parameter for tuning is only the variance. Compared with algorithms above, the advantage of this algorithm does not need many trials to tune parameters. The algorithm is auto-adaptive based on the structure of its algorithm. It is difficult to implement the algorithm.

Answers

Algorithm Steps

1. denoised pixels in the patches in 2-D dimension (this can be done by trivial algorithm)

1. match patches of pixels to the reference pixel to group.
2. transform these groups to other space and use hard thresholding to denoise the groups and then transform them back.
3. the referenced patch of the referenced pixel can be a member of several groups, so the weight function is designed to weight these estimation source. There are two type of patch. The patch is in the group of the referenced pixel and the patch in the other group but referencing the pixel. The weight is given to pixels in these patches. The detailed formula is shown in formula.
4. estimate the reference pixel value from all these pixels in all these groups.
5. repeat the above step by using Wiener filtering again. [4]

Even for neutral network, the BM3D is not beat.

Implementation

The implementation is based on the matlab framework [2]. Figure 2.4.1 and 2.4.2 shows result of using BM3D algorithm.

(e). Mixed noises in color image (10%)

Motivation

There is no specific code for this problem. Hence, all parts in this problem is to analyze the above denoising algorithm and compare their advantages and disadvantages. There is no panacea algorithm to denoise the image. The limit of the algorithm comes from its computation procedure. Almost all algorithm mainly focus on the neighboring pixels. In contrast, BM3D algorithm uses the global information to estimate the noisy pixel. The advantage of BM3D is that it does not need much effort to tune parameter but it can get good performance even with first trial.

Approach and procedure

The first step is to analyze the image with eye to decide the type and possible mixed noise that may be included in the image. The first step is to use median filter to filter the impulse noise and then use BM3D algorithm to filter white noise.

Results

To handle two type of noise, median filter can be chosen to handle impulse noise. There are many algorithms to handle the gaussian noise in three channels. More mathematical tool can be used to analyze the image character itself like its histogram or other estimation method. If more detailed information can be known, the specific chosen algorithm will perform better and the tuning procedure will be efficient.

Discussion

Compared with the original image, the noisy image is full of pure black, yellow peak value pixel. From naked eye observation, this type of noise is impulse pepper. And the noise distribution for the value seems uniformly. Hence, there is other gaussian noise in the image.

Answers

1. two type of noise is added to the image, impulse noise and uniform noise (Gaussian noise).

2. The impulse noise should be filtered firstly by median filter algorithm. If it is filtered the average filter firstly, the abnormal value will be distributed equally to all pixels in the window. Hence, the following median filter is useless. It cannot detect the peak pixel in the window. The other type of filter for while noise should be used next.

3. The median filter is chosen to eliminate or alleviate the impulse pepper noise. The BM3D algorithm is chosen to filter gaussian noise. The BM3D has been tested for long time and there is no other algorithm that can beat it. Hence, it is unnecessary to waste time to use other algorithm for gaussian noise. Also, the other algorithm needs great efforts to tune the parameters.

Appendix A

1. Compilation environment

Compilation environment is Visual Studio 2019

2. Coding IDE

IDE is also Visual Studio 2019

3. Github code cloud store

The code is uploaded to the github of my repository for back-up

The github access link is https://github.com/thetimeofblack/EE569-Digital-Signal-Processing.git There are more temporary image and code in the repository.

Reference and Bibliography

[1] P. Gong, "Histogram-Based Operation," 2020. [source]. Available: https://nature.berkeley.edu/~penggong/textbook/chapter6/html/sect61.htm. [visit date: 2020].

[2] .. Kostadin, 2007. [source]. Available: https://www.cs.tut.fi/~foi/GCF-BM3D/BM3D_TIP_2007.pdf.

[3] Y. Mäkinen.etc, "Image and video denoising by sparse 3D transform-domain collaborative filtering," 2007. [source]. Available: http://www.cs.tut.fi/~foi/GCF-BM3D/.

[4] HCI / Heidelberg University Prof. Fred Hamprecht, "2.4 BM3D for Image Denoising | Image Analysis Class 2013," 26 4 2013. [source]. Available: https://www.youtube.com/watch?v=BlDl6M0go-c. [visit date: 27 1 2019].